3-14-2007

Quick-Turn Finite Element Analysis for Plug-and Play Satellite Structures

Jeffrey E. Naff

QUICK-TURN FINITE ELEMENT ANALYSIS FOR PLUG-AND-PLAY SATELLITE STRUCTURES

THESIS

Jeffrey E Naff, Captain, USAF

AFIT/GA/ENY/07-M15

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GA/ENY/07-M15

QUICK-TURN FINITE ELEMENT ANALYSIS FOR PLUG-AND-PLAY SATELLITE STRUCTURES

THESIS

Presented to the Faculty

Department of Aeronautics and Astronautics

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Astronautical Engineering

Jeffrey E. Naff, BS

Captain, USAF

March 2007

AFIT/GA/ENY/07-M15

QUICK-TURN FINITE ELEMENT ANALYSIS FOR PLUG-AND-PLAY SATELLITE STRUCTURES

Jeffrey E. Naff, BS
Captain, USAF

Approved:

Signed	*14 March 2007*
Maj. Eric D. Swenson, Ph.D (Chairman)	Date
Signed	*14 March 2007*
Richard G. Cobb, Ph.D (Member)	Date
Signed	*14 March 2007*
Robert A. Canfield, Ph.D (Member)	Date

Abstract

Plug-and-play (PnP) satellite construction is a key component of the US Air Force Operational Responsive Space (ORS) effort. The goal of ORS is to provide mission specific satellite support by configuring and launching a satellite to a selected orbit within days of the request. One major challenge during the time limited process is to accurately predict the response of the satellite to harmonic loads that occur during launch and satellite operation. Given the time limitations, constructing finite element (FE) models by traditional methods is not currently a viable option for the ORS timeline. By implementing an approach for rapid FE model creation, we can significantly reduce the timeline from weeks to hours. The advantages to our approach include simplification of model creation, ease of design modifications, and significant reduction in the FE model creation timeline; all lending this approach for utilization within the ORS acquisition cycle.

Table of Contents

List of Figures

List of Tables

QUICK-TURN FINITE ELEMENT ANALYSIS FOR
PLUG-AND-PLAY SATELLITE STRUCTURES

1 Introduction

The US Air Force has initiated programmatic efforts to revolutionize the development of space assets. For years, the acquisition of US Air Force space resources has severely lagged behind their operational need. Satellite developmental timelines of two to three years for each asset is unacceptable to meet the overlying needs of the Air Force [8]. Therefore, the concept of modifying the acquisition cycle to accommodate immediate reaction to stated needs is a must.

To achieve a space initiative that responds quickly to user need, a paradigm shift must occur within the space industry. We know that the development of systems and their timelines are greatly affected by the number of assets planned and the availability of existing technologies utilized within each asset. The acquisition cycle of satellite development can be strategically altered to better reflect these development limiting factors. The Operational Responsive Space (ORS) program is undertaking the task of altering the development model to allow space assets to be available when the mission requires it.

1.1 Operational Responsive Space

The effort to develop an ORS program has been in existence for many years. Many US government organizations are involved in the project, to include the Air Force Research Laboratory (AFRL), the National Reconnaissance Organization (NRO), National Air and Space Administration (NASA), and many other Department of Defense (DoD) organizations.

The primary goal has been to demonstrate the ability to produce and launch a time sensitive satellite with a rapid development cycle. The first successful launch was TACSAT

2, launched in December 2006, followed by the Launch of TACSAT 1 in January 2007. Each of these missions shared the same development map of twelve months between mission initiation and launch. The TACSAT missions confirmed the belief that given the advances in technology and the availability of Commercial-Off-The-Shelf (COTS) components, a quick-delivery spacecraft is within grasp.

While the construction of the TACSAT programs has seen success, it also has fallen victim to the standard acquisition pitfalls that affect all programs. TACSAT 1 launch has slipped over a year and a half from initial schedule, bringing the timeline to that of a standard acquisition cycle. The future goals of the ORS program detail the need to provide mission specific satellite support by configuring and launching a satellite to a selected orbit within days, not years, of the request. To achieve this, a new acquisition cycle has been developed that re-organizes the key components of the standard cycle to provide an acquisition environment that allows the complete construction of individual elements without final declaration of mission intent.

Current space system acquisition maintains a strict step-by-step procedure for the development of systems. The first step is centered on mission requirements; everything must be defined prior to any further activity. Upon completion, system design is initiated. The majority of the work is achieved with a stove-pipe mentality; all work is completed on a component prior to any system level integration activity. After the majority of the integration has occurred, typically around eighty percent, final assembly and test are initiated. At the conclusion of all testing activities, the satellite is launched, completing the two to three year cycle. The comparison between the standard acquisition cycle and the proposed ORS cycle is demonstrated in Figure 1.1.

Many differences can be seen between the proposed ORS cycle and existing satellite procurement approaches. To begin with, the first step within the ORS cycle is not mission requirement development, but an overall definition of the interface for the spacecraft. Following that, the component level design begins, which is closely tracked by the system integration step. When it is ensured that all components are fully integrated, they are placed in storage to await a need within a mission.

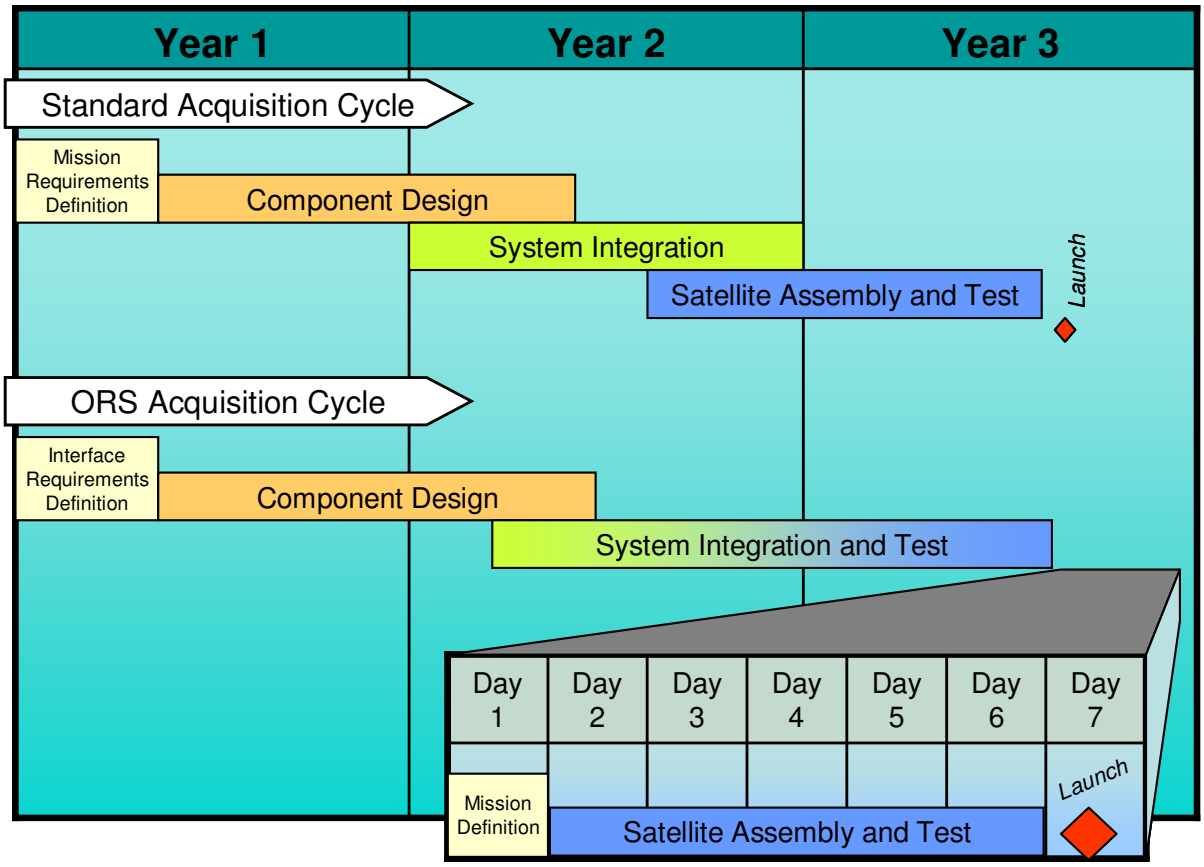


Figure 1.1: Comparison of Acquisition Model Timelines

The final process is the key to a successful responsive system. At mission call up, a seven day window opens and the final activities begin. The mission requirements are analyzed and the specific components required for the given mission are selected. For example, if the mission is reconnaissance oriented, a group of components dealing with imaging or surveillance would be selected. The spacecraft design is then finalized, assembled, and tested prior to launch. On the seventh and final day, the satellite is launched to the required orbit.

As can be seen, the ORS conceptual acquisition plan redefines the paradigm for spacecraft fabrication by dictating a two-phase effort. The first, defined as the component development and integration stage, provides the ability to design and integrate a multitude of components for all types of missions. With the given integration concept, each specific component will have the ability to communicate with the designated segments, independent

of the location or bus attachment of the device. Included in this stage of development is all interoperational testing to assure all systems readily communicate with the associated applications. With this model in place, all components and the associated busses would reach full system maturation prior to any assembly activities.

The second stage, or satellite assembly stage, is dedicated to the development of a given satellite by utilizing the previously produced components. The overall design of the satellite will be dependent on the needs of the mission and will not be completed until the assembly stage. Once the design is finalized, the overall assembly and test of the vehicle is initiated. Because all basic communication, interoperability, and system effectiveness testing will have been completed during the system integration stage, only pre-flight checks will be required for the satellite's hardware. The remaining design element that is incomplete at this point is the structure: the definition of how the hardware will be contained for operational use. Because the structure of the satellite will differ for every iteration and will not be defined until the assembly stage, all standard vibration and thermal testing will be accomplished within this stage. The development of a finite element (FE) model of the finalized design would be valuable in preventing errors within the assembly stage testing. However, due to the unknown quantity of potential satellite configurations, modeling of each system prior to this stage is not a feasible option.

1.2 Plug-and-Play Satellite Development

The major concept that provides the ORS program with the ability to construct a satellite within days of mission call-up is the utilization of Plug-and-Play (PnP) components. The PnP model's defining characteristics are the development of all system equipment (both hardware and software) and the completion of all integration and interoperability testing prior to mission design. The remaining activities to be completed prior to launch are then limited to system assembly and checkout.

The ORS program acquisition model presented earlier requires that all development and integration activities occur prior to availability of any mission requirements. Therefore, many assumptions must be made with respect to anticipated satellite capability. In order for the PnP concept to be effective for most missions, the development of 'capability' packages,

or mission kits, must be accomplished. For instance, a sensor group would be specifically designed for imaging support, like infrared or visuals. Likewise, a communication group would be another package. The individual modules within a specified package are inherently interoperable within that package. The major system wide integration effort between each of the capability packages will occur within the integration stage of the ORS cycle. The ORS concept is demonstrated in Figure 1.2.

The development of the satellite bus will be a vastly different effort than in previous non-PnP development programs. The intrinsic capabilities of a PnP system allow for the addition or subtraction of any properly formatted device at any available connection point. To present a relevant example of this capability, we examine the current Universal Serial Bus (USB) used within the personal computing markets. The USB standard allows a processing unit to connect with any peripheral device that shares the same USB standard. While the current USB version 2.0 is not suitable for the space environment or the amount of data transfer, a more robust update will be instrumental to the success of the PnP concept.

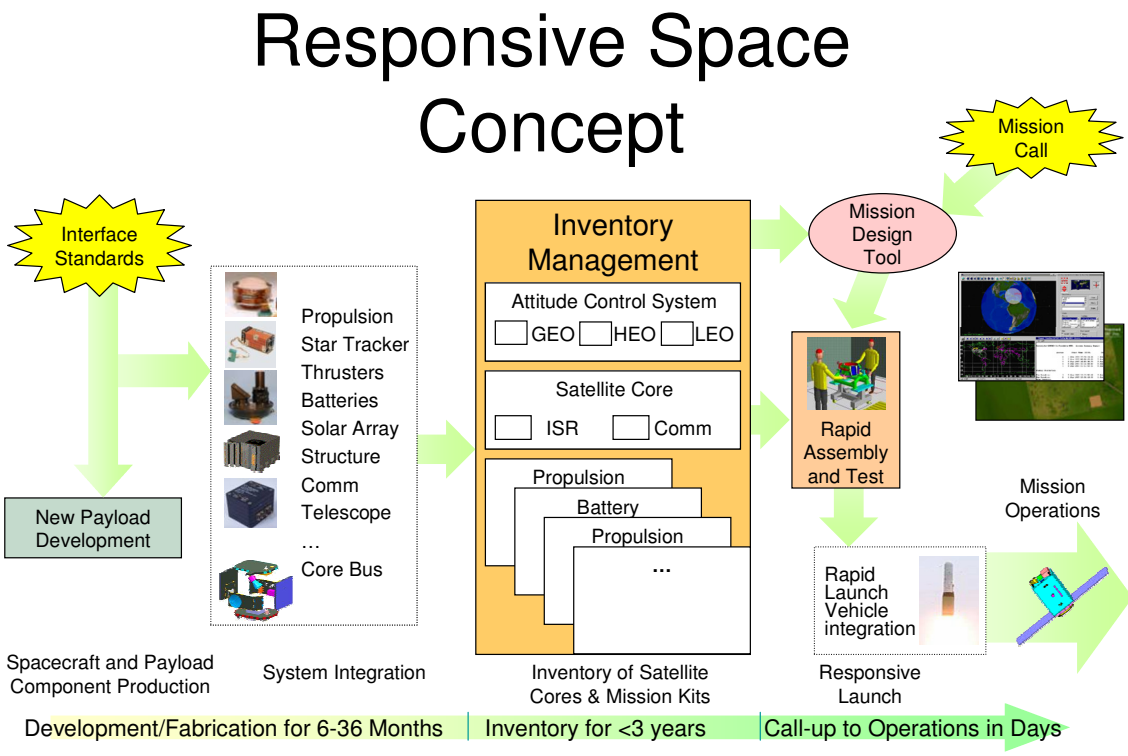


Figure 1.2: Responsive Space Plug-and-Play Concept [14]

While the PnP model is suited for the sensor packages and the spacecraft bus, the application of the PnP model to the satellite structure is a key component for overall success. Each satellite produced by the ORS effort will be different, and therefore each structure's configuration will differ. For instance, a dedicated surveillance satellite will have different power and envelope requirements than a communications satellite, and therefore the structures will differ. Due to the time limit imposed, the development of a proprietary, spacecraft specific structure is not feasible. Therefore, all structural components must be designed and developed within the component development stage of the ORS cycle. If the mentality that governs sensor development is directly applied to the structure, development of individual parts can occur while the final design of the system is not complete until the assembly stage. A generic list of structural parts can be developed and prepped for use prior to the assembly stage, including varying sizes of structural beams, panels, and attachment points. At the point of mission definition, the selected configuration would then dictate the structural arrangement and design.

1.3 System Modeling within the PnP Concept

One aspect of spacecraft development that does not readily fit within the ORS paradigm is modeling and simulation (M&S) of the complete satellite. Within a standard acquisition cycle, computer simulation of a product can significantly reduce the cost and time of design and test cycle, but typically takes between two and four months to produce a validated, accurate model. Given the time available for satellite assembly, the standard approach for computer simulation cannot be applied within the ORS model.

Two questions arise with regards to M&S from the limitation applied to the simulation development. First, and foremost, "Can we eliminate the requirement to simulate the assembled vehicle?" The quick answer is no. Two major testing applications benefit from the use of M&S, the response from harmonic loading (vibration testing) and the effect that extreme heat and cold will have on the system (thermal testing). All testing with respect to these two areas must be completed on the assembled structure prior to launch. By creating a computer simulation of the satellite, we can accurately predict the system responses and

modify the design if necessary, all prior to actual construction and testing. If the simulation effort does not occur, the satellite could fall into a test-fix-retest cycle that can easily drive a system wide delay.

The second question, "Can we complete the M&S effort prior to mission definition?", again is answered with no. Due to the nature of the ORS effort, the large number of potential configurations drives the structural design to change with each asset produced. Development of many models that can accurately simulate every configuration produced is not practical.

An approach evaluated in this research is the development of accurate and validated models of each component available for use and assembling the models after mission definition, commonly referred to as a substructuring approach. Substructuring is a common practice in the automotive and aerospace industries [3]. We apply the substructuring approach to the structural components of a satellite.

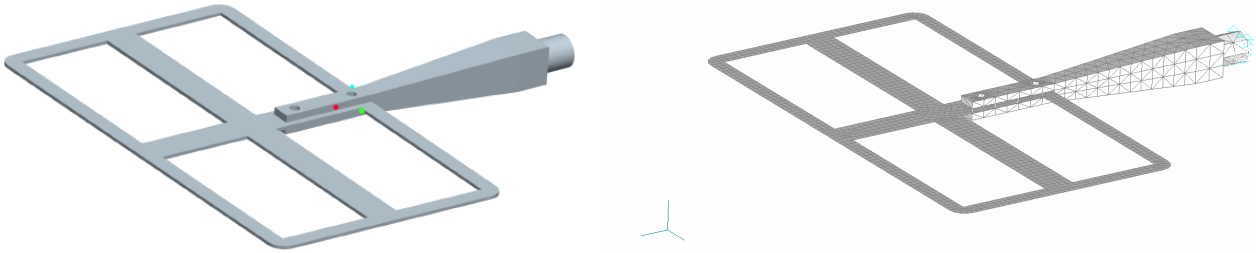
The development of FE models requires experienced manpower to properly ensure the entire model, whether substructures are used or not, is correctly assembled. If correction must occur within the model after it has been completed, the attachment of each element must be rechecked within the modified sections; otherwise further errors may be introduced. It is common practice to simply create a generic model with slight additions or subtractions based on the unit being tested. This approach is not applicable to the ORS as the creation of a given satellite FE model does not validate all of the potential variations than can exist within the ORS approach.

1.4 Industry Standard Practices for Model Creation

The development of FE models is significantly eased by utilizing the computer based object drawing and analysis applications, also known as Computer Automated Engineering (CAE). The most common FE analysis solver is Nastran; developed by the MacNeal Schwindler Corporation (MSC) in 1968 for use by the NASA for spacecraft development. Today, Nastran is the solver for many popular FE software titles. Many other companies have developed proprietary FE analysis code that does not utilize the NASTRAN engine, but

provides equivalent results: i.e. Abaqus, Pro/Mechanica, Structural Dynamics Toolbox for MATLAB (SDT), etc.

The advantages of computer-based analysis deal with the ability to perform significant numbers of calculations quickly, predicting the behavior of a structure. It is not uncommon to have FE models within the automotive and aeronautical industry that have more than several million Degrees of Freedom (DOF). FE analysis associated with such large models requires significant computing power to complete in a reasonable amount if time. Figure 1.3(a) demonstrates a component developed with a solid modeling application and Figure 1.3(b) shows the associated FE mesh.



(a) (b)

Figure 1.3: Computer Developed Model (a) Solid Model, (b) FE Model

1.5 Utilization of Substructures within Model Development

The use of substructures within a complex model is common practice in modern FE model development. The best example for substructure models is the classic airplane model. Two methods can be utilized in construction of a FE model for the structure of the aircraft. The first is to mesh the entire structure in one complete model and the second is to break the model into distinct, separate sub-models, or substructures, and create an element mesh for the individual sections. The separate meshes are later connected at their respective interfaces.

The development of substructure models can be extremely advantageous. By breaking a structure into substructures, or superelements, typically associated with development

departments (i.e. wing, fuselage, tail section, and landing gear), the individual FE models can be created and refined independently of the complete model. It is ideal to have the FE models of each substructure created by the specific component designers, ensuring that areas of specific analytical concern can be effectively inserted into the model.

Application of the substructuring concept can also reduce the complexity of analysis completed on the entire models. Due to independent model development by the specified design team, the verification and validation of the model to the actual structure can occur prior to inclusion within the overarching model, lending a higher level of confidence of the data received through analysis. With accurate data in hand, design and deployment decisions can be simplified and faster response can occur.

1.6 Issues Associated with Patchwork Solutions

While the substructure approach has been widely adapted, the compilation of the models into a final, complete model is still a concern. Traditionally, when FE models are created independently of each other, the elemental meshes are considerably different, providing a complicated task of stitching the meshes together at their respective interfaces, commonly referred to as a patchwork solution. Each model produced has a series of nodes that are specifically designated as connection nodes. The connection points between the models can insert a large quantity of unknown vibratory reactions and if not properly meshed together, the validation of the entire model is placed at risk. The method presented in this paper evaluates an approach for automating the process of compiling the complete FE model by standardizing the satellite model interfaces, eliminating the inconsistency at the model connection points and the requirement for a patchwork solution.

2 Finite Element Analysis Methodology

In order to rapidly and accurately develop a finite element (FE) model to predict the response of a complete satellite within the Operational Responsive Space (ORS) program concept, we must first discuss the fundamentals of FE modeling. In this chapter, we will look into the current methods utilized within industry. Mathematical development of the mass and global stiffness matrices is the basic starting point for any FE model development. The similarities of the approach presented in this chapter apply equally to both the mass and stiffness matrices; we will only discuss the global stiffness matrix (K). The development of basic FE models is widely published; if more information is needed, any introductory FE analysis book can be referenced.

Standard FE analysis and model development give us a method to develop accurate results for a modeled system. However, the standard methods do not provide the ability to develop and analyze a system quickly. In fact, the standard methods are quite time consuming. To demonstrate quick FE model assembly, we will present a method that utilizes simplistic models but develops construction guidelines that can be applied to more complex structures. The following equation development is extracted from Tirupathi Chandrupatla's textbook "Introduction to Finite Elements in Engineering." [1]

2.1 Basic Finite Element Analysis

The development of the global stiffness matrix (K) and the mass matrix (M), the key components of FE analysis, requires us to examine the basic attributes of an elastic body. To begin the analysis, we utilize the elementary beam equations for stress (σ), strain (ε), and the deflection of the centroidal axis (v). The axial stress equations are functions of the bending moment (M) and the moment of inertia (I). Figure 2.1 depicts the graphical representation of the elementary stress field for a beam.

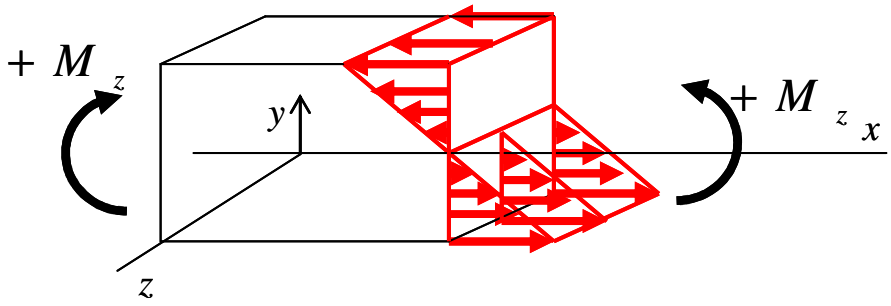


Figure 2.1: Stress Field for a Beam in Bending

$$\sigma_x = -\frac{M_z}{I_z} y \quad \text{or} \quad \sigma_x = -\frac{M_y}{I_y} z \tag{2.1}$$

$$\varepsilon = \frac{\sigma}{E} \tag{2.2}$$

$$\frac{d^2 v}{dx^2} = \frac{M}{EI} \tag{2.3}$$

The potential energy of the beam, Π, is defined by Equation 2.4 where u is defined as the displacement of a given point, f is the distributed force per unit volume, T is the surface traction, and P_i is a load acting at point i.

$$\Pi = \frac{1}{2}\int_V \sigma^T \varepsilon \, dV - \int_V u^T f \, dV - \int_S u^T T \, dS - \sum_i u_i^T P_i \tag{2.4}$$

Equation 2.2 can then be broken into two segments, Strain Energy (U) and Work Potential (WP).

$$\Pi = U + WP \tag{2.5}$$

$$U = \frac{1}{2}\int_V \sigma^T \varepsilon \, dV \tag{2.6}$$

$$WP = -\int_V u^T f \, dV - \int_S u^T T \, dS - \sum_i u_i^T P_i \tag{2.7}$$

When developing the stiffness matrix for a given element, we only use the strain energy portion of the potential energy equation. The work potential is used for applied forces and has no effect on the element's stiffness.

The volume within Equation 2.6 can be broken out to determine the strain energy of an element with a fixed length *dx*.

$$\begin{aligned} dU &= \frac{1}{2}\int_A \sigma^T \varepsilon \, dA dx \\ &= \frac{1}{2}\left(\frac{M^2}{EI^2}\int_A y^2 dA\right)dx \end{aligned} \tag{2.8}$$

Equation 2.8 can then simplified further by the utilizing the equation for the moment of inertia (Equation 2.9) which further reduces to Equation 2.10.

$$I = \int_A y^2 dA \tag{2.9}$$

$$dU = \frac{1}{2}\frac{M^2}{EI}dx \tag{2.10}$$

By using the relation developed in Equation 2.3, the total strain energy of the beam is derived.

$$U = \frac{1}{2}\int_0^L EI\left(\frac{d^2 v}{dx^2}\right)^2 dx \tag{2.11}$$

The Hermite shape functions used for beam analysis give us an equation for the deflection of the centroidal axis (*v*), while looking at nodal displacement and slope.

$$v = H_1 v_1 + H_2\left(\frac{dv}{d\xi}\right)_1 + H_3 v_2 + H_4\left(\frac{dv}{d\xi}\right)_2 \tag{2.12}$$

The coordinate transformation gives the relationship between x and ξ. The length of the element, l, is defined as the distance between the node locations, x_1 and x_2.

$$x = \frac{x_1 + x_2}{2} + \frac{x_2 - x_1}{2}\xi \tag{2.13}$$

where

$$dx = \frac{x_2 - x_1}{2} d\xi = \frac{l}{2} d\xi \tag{2.14}$$

By using the chain rule and Equation 2.14, we transform the slope of v to be valid in terms of x.

$$\frac{dv}{d\xi} = \frac{dv}{dx}\frac{dx}{d\xi} = \frac{l}{2}\frac{dv}{dx} \tag{2.15}$$

By rewriting Equation 2.12 with inclusion of the nodal displacement vector $\boldsymbol{q}$, we can simplify the equation as follows.

$$v = H_1 q_1 + H_2 \frac{l}{2} q_2 + H_3 q_3 + H_4 \frac{l}{2} q_3 \tag{2.17}$$

$$v = Hq \tag{2.18}$$

$$H = \begin{bmatrix} H_1 & \frac{l}{2}H_2 & H_3 & \frac{l}{2}H_4 \end{bmatrix}$$

$$q = \begin{bmatrix} \underbrace{\underbrace{q_1}_{Translation} \quad \underbrace{q_2}_{Rotation}}_{Node\,1} & \underbrace{\underbrace{q_3}_{Translation} \quad \underbrace{q_4}_{Rotation}}_{Node\,2} \end{bmatrix}^T \tag{2.19}$$

Given the relationship in Equation 2.15, we can expand it to incorporate Equation 2.18.

$$\frac{dv}{dx} = \frac{2}{l}\frac{dv}{d\xi}$$
$$\frac{d^2v}{dx^2} = \frac{4}{l^2}\frac{d^2v}{d\xi^2} = \frac{4}{l^2}\frac{d^2H}{d\xi^2}q \tag{2.20}$$

By substituting Equation 2.20 into Equation 2.11, we get an equation for strain that utilizes the shape functions.

$$U = \frac{1}{2}\int_0^l EI\left(\frac{4}{l^2}\frac{d^2H}{d\xi^2}q\right)^2 dx \tag{2.21}$$

$$U = \frac{1}{2}EI\int_{-1}^{+1}\frac{16}{l^4}q^T\left(\frac{d^2H}{d\xi^2}\right)^T\left(\frac{d^2H}{d\xi^2}\right)q\frac{l}{2}d\xi \tag{2.22}$$

$$\frac{d^2H}{d\xi^2} = \left[\frac{3}{2}\xi \quad \left(\frac{(3\xi-1)l}{4}\right) \quad -\frac{3}{2}\xi \quad \left(\frac{(3\xi+1)l}{4}\right)\right] \tag{2.23}$$

$$\int_{-1}^{+1}\xi^2 d\xi = \frac{2}{3}$$
$$\int_{-1}^{+1}\xi d\xi = 0 \tag{2.24}$$
$$\int_{-1}^{+1}d\xi = 2$$

When Equation 2.22 is substituted into Equation 2.21 and the relationships in Equation 2.24 are applied, the potential can be compactly written as Equation 2.25.

$$U = \frac{1}{2}q^T\frac{EI}{l^3}\begin{bmatrix} 12 & 6l & -12 & 6l \\ 6l & 4l^2 & -6l & 2l^2 \\ -12 & -6l & 12 & -6l \\ 6l & 2l^2 & -6l & 4l^2 \end{bmatrix}q \tag{2.25}$$

The potential energy equation is then simplified and k, the elemental stiffness matrix, is defined.

$$U = \frac{1}{2} q^T k q \tag{2.26}$$

$$k = \frac{EI}{l^3} \begin{bmatrix} 12 & 6l & -12 & 6l \\ 6l & 4l^2 & -6l & 2l^2 \\ -12 & -6l & 12 & -6l \\ 6l & 2l^2 & -6l & 4l^2 \end{bmatrix} \tag{2.27}$$

The development of the elemental stiffness matrix *k* has been derived using a two-dimensional (2D) beam. While utilizing a 2D beam is ideal for initial equation development, we must broaden the equations to represent the three-dimensional (3D) environment for realistic modeling of the beam.

2.1.1 Extension to Three-Dimension Beams

The major difference between the two-deminsional beam used previously and the ideal three dimensional beam element used from this point forward are the rotations allowed in each axis at each node and the additional third axis translation. The categorization frame is simply a beam that has a given degree of freedom that is designated as rotation. Therefore, a 3D frame has six degrees of freedom (DOF) as seen in Table 2.1.

Table 2.1: Three-Dimensional Frame DOF Definition

DOF Number	Node Relation
1	X-Direction Translation
2	Y-Direction Translation
3	Z-Direction Translation
4	Rotation about X Axis
5	Rotation about Y Axis
6	Rotation about Z Axis

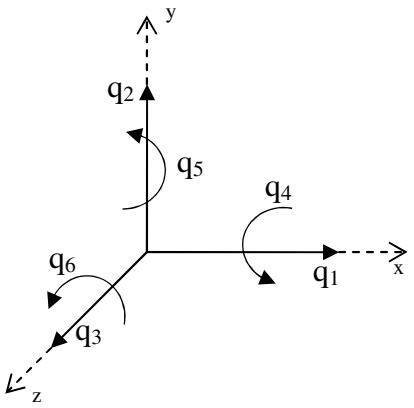


Figure 2.2: Six Degrees-of-Freedom

Given that all six DOFs will be used, the nodal displacement vector $\boldsymbol{q}$ for a two-node beam element will have twelve components; six generalized displacements per node with three each for translation and rotation, as represented in Figure 2.2.

$$\boldsymbol{q} = \left[\underbrace{\underbrace{q_1 \quad q_2 \quad q_3}_{Translation} \quad \underbrace{q_4 \quad q_5 \quad q_6}_{Rotation}}_{node\ 1} \quad \underbrace{\underbrace{q_7 \quad q_8 \quad q_9}_{Translation} \quad \underbrace{q_{10} \quad q_{11} \quad q_{12}}_{Rotation}}_{node\ 2} \right]^T \tag{2.28}$$

The elemental stiffness matrix (k) for the three dimensional beam will be an extension of Equation 2.27. The main difference will be the inclusion of the axial (AS) and shear (TS) terms. There are six generic equations needed to fill the stiffness matrix. The quantity GJ is representative of the torsional stiffness. G is the shear modulus and J is the polar moment of inertia.

$$AS = \frac{EA}{l} \tag{2.27}$$

$$TS = \frac{GJ}{l} \tag{2.28}$$

The constants a_i, b_i, c_i, and d_i are defined as follows.

$$a_i = 12\frac{EI_i}{l^3}, \qquad i = z, y \tag{2.29}$$

$$b_i = 6\frac{EI_i}{l^2} \qquad i = z, y \tag{2.30}$$

$$c_i = 4\frac{EI_i}{l}, \qquad i = z, y \tag{2.31}$$

$$d_i = 2\frac{EI_i}{l'}, \qquad i = z, y \tag{2.32}$$

Now, the assembly of the elemental stiffness matrix k is a straight forward use of the previous equations.

$$k' = \begin{bmatrix}
AS & 0 & 0 & 0 & 0 & 0 & -AS & 0 & 0 & 0 & 0 & 0 \\
 & a_z & 0 & 0 & 0 & b_z & 0 & -a_z & 0 & 0 & 0 & b_z \\
 & & a_y & 0 & -b_y & 0 & 0 & 0 & -a_y & 0 & -b_y & 0 \\
S & & & TS & 0 & 0 & 0 & 0 & 0 & -TS & 0 & 0 \\
 & y & & & c_y & 0 & 0 & 0 & b_y & 0 & d_y & 0 \\
 & & m & & & c_z & 0 & -b_z & 0 & 0 & 0 & d_z \\
 & & & m & & & AS & 0 & 0 & 0 & 0 & 0 \\
 & & & & e & & & a_z & 0 & 0 & 0 & -b_z \\
 & & & & & t & & & a_y & 0 & b_y & 0 \\
 & & & & & & r & & & TS & 0 & 0 \\
 & & & & & & & i & & & c_y & 0 \\
 & & & & & & & & c & & & c_z
\end{bmatrix} \tag{2.33}$$

The final step for the elemental stiffness matrix *k'* development is to perform a coordinate transfer from local to global coordinates. The local stiffness matrix *k'* transformation is projected on the global coordinate system by use of a transformation matrix *L*. *L* is composed of a series of direction cosine matrices (λ), where l_i, m_i, and n_i are the cosines between the elemental i-axis (i.e. i = x, y, z) and the global *x*, *y*, and *z* axis respectively.

$$\lambda = \begin{bmatrix} l_1 & m_1 & n_1 \\ l_2 & m_2 & n_2 \\ l_3 & m_3 & n_3 \end{bmatrix} \tag{2.34}$$

Because λ is a 3x3 matrix, it is repeated along the diagonal of *L* until the proper dimension is attained. For the two-node, twelve-DOF element expressed in Equation 2.33, λ is repeated four times, ensuring the dimension of the transformation matrix *L* matches the dimension of *k*. *L* is then filled with zeros, represented in Equation 2.35 as **0**.

$$L = \begin{bmatrix} \lambda & & & \mathbf{0} \\ & \lambda & & \\ & & \lambda & \\ \mathbf{0} & & & \lambda \end{bmatrix} \tag{2.35}$$

The elemental displacement vector (***q'***) and the elemental stiffness matrix (*k'*) are defined in the global coordinates as follows.

$$\boldsymbol{q}' = L\boldsymbol{q} \tag{2.36}$$

$$k = L^T k' L \tag{2.37}$$

2.1.2 Assembly of the Global Stiffness Matrix (K)

After the elemental stiffness matrices k_j have been created and transformed to the global coordinate system, we place them in the global stiffness matrix (K). Figure 2.3 shows an example system we will use to discuss the assembly process. Both element 1 and 2 have 12 DOF, and the global stiffness matrix has a total DOF of 18, because both elements share node 2. The dimension of the global stiffness matrix K is the number of nodes multiplied by six.

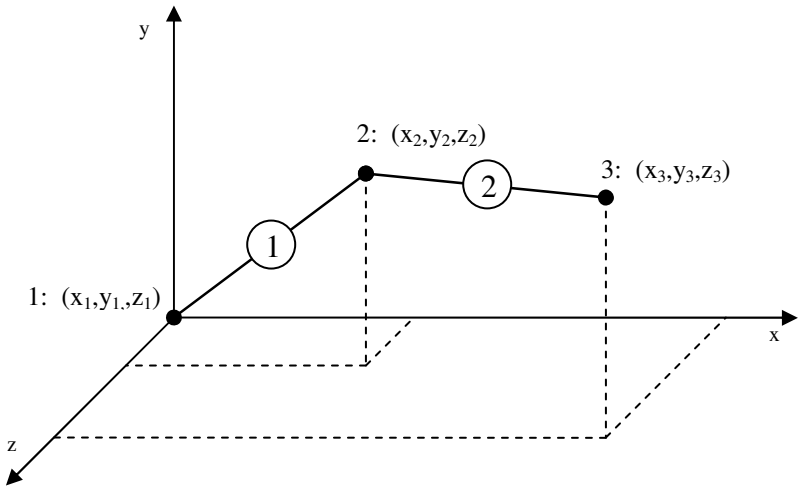


Figure 2.3: Three-Dimensional Two Element System

We begin by calculating the elemental stiffness k' of each of the two elements in their local coordinate systems. Equation 2.38 shows the k' for each member where j represents the member number, 1 or 2. Note that the stiffness matrices are symmetric about the diagonal. We will continue to utilize the symmetry throughout the paper, but will eliminate the designation.

$$k'_j = \begin{bmatrix} AS_j & 0 & 0 & 0 & 0 & 0 & -AS_j & 0 & 0 & 0 & 0 & 0 \\ & a_{zj} & 0 & 0 & 0 & b_{zj} & 0 & -a_{zj} & 0 & 0 & 0 & b_{zj} \\ & & a_{yj} & 0 & -b_{yj} & 0 & 0 & 0 & -a_{yj} & 0 & -b_{yj} & 0 \\ S & & & TS_j & 0 & 0 & 0 & 0 & 0 & -TS_j & 0 & 0 \\ & y & & & c_{yj} & 0 & 0 & 0 & b_{yj} & 0 & d_{yj} & 0 \\ & & m & & & c_{zj} & 0 & -b_{zj} & 0 & 0 & 0 & d_{zj} \\ & & & m & & & AS_j & 0 & 0 & 0 & 0 & 0 \\ & & & & e & & & a_{zj} & 0 & 0 & 0 & -b_{zj} \\ & & & & & t & & & a_{yj} & 0 & b_{yj} & 0 \\ & & & & & & r & & & TS_j & 0 & 0 \\ & & & & & & & i & & & c_{yj} & 0 \\ & & & & & & & & c & & & c_{zj} \end{bmatrix} \quad (2.38)$$

The global stiffness matrix *K* is a combination of both element stiffness matrices, after transformation to the local coordinate system using Equation 2.37. Equation 2.39 shows the placement of each element's transformed *k* matrix, demonstrating the overlap between each element due to the sharing of node 2 between both elements.

$$\boldsymbol{K} = \begin{bmatrix} k_1 & \mathbf{0} \\ & \text{Equation 2.34} \\ & k_2 \end{bmatrix} \quad (2.39)$$

The submatrix specific to node 2 is highlighted in Equation 2.40. Because the elements share this node, the 6x6 matrix that overlaps is the sum of k_1 and k_2 at those DOF.

$$K_{7\to12}=\begin{bmatrix} AS_1+AS_2 & 0 & 0 & 0 & 0 & 0 \\ & a_{z1}+a_{z2} & 0 & 0 & 0 & b_{z2}-b_{z1} \\ & & a_{y1}+a_{y2} & 0 & b_{y1}-b_{y2} & 0 \\ & & & TS_1+TS_2 & 0 & 0 \\ & & & & c_{y1}+c_{y2} & 0 \\ & & & & & c_{z1}+c_{z2} \end{bmatrix} \tag{2.40}$$

2.2 Basics of Nodal Segregation

To automate the process of assembling the global stiffness matrix from the component stiffness matrices, we need to define a procedure that easily places the component matrices into the global stiffness matrix but maintains the individual component characteristics. By recognizing that the component stiffness matrix k_c can be viewed as a compilation of specific nodal information and the associated connectivity information, we can break k_c into submatrices; nodal and connectivity, as demonstrated by the single element matrix in Equation 2.41.

$$k_c=\left[\begin{array}{cccccc|cccccc} \overbrace{AS & 0 & 0 & 0 & 0 & 0}^{Node\ 1} & \overbrace{-AS & 0 & 0 & 0 & 0 & 0}^{Connectivity\ 1\to2} \\ & a_z & 0 & 0 & 0 & b_z & 0 & -a_z & 0 & 0 & 0 & -b_z \\ & & a_y & 0 & -b_y & 0 & 0 & 0 & -a_y & 0 & -b_y & 0 \\ & & & TS & 0 & 0 & 0 & 0 & 0 & -TS & 0 & 0 \\ & & & & c_y & 0 & 0 & 0 & b_y & 0 & d_y & 0 \\ & & & & & c_z & 0 & -b_z & 0 & 0 & 0 & d_z \\ \hline & & & & & & AS & 0 & 0 & 0 & 0 & 0 \\ & & & & & & & a_z & 0 & 0 & 0 & -b_z \\ & & & & & & & & a_y & 0 & b_y & 0 \\ & & & & & & & & & TS & 0 & 0 \\ & & & & & & & & & & c_y & 0 \\ & & & & & & & & & & & c_z \end{array}\right] \tag{2.41}$$

To demonstrate the submatrix separation in a more complex system, we will examine the two-element, three-node example shown in Figure 2.3. From the applicable global stiffness matrix *K* for this system, Equation 2.38, and the graphical drawing, we can produce an element connectivity diagram which will help with the separation of nodes within K.

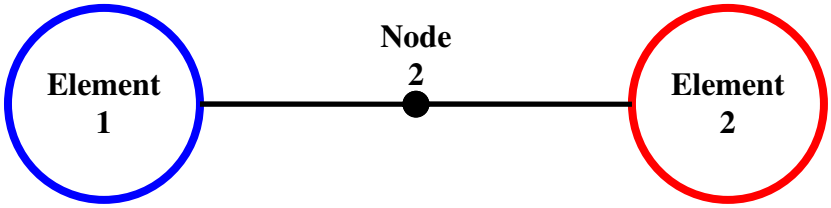


Figure 2.4: Connectivity Diagram for a Two-Element System

Figure 2.4, shows element 1 and 2 are connected by node 2. The global stiffness matrix *K* is shown in Figure 2.5.

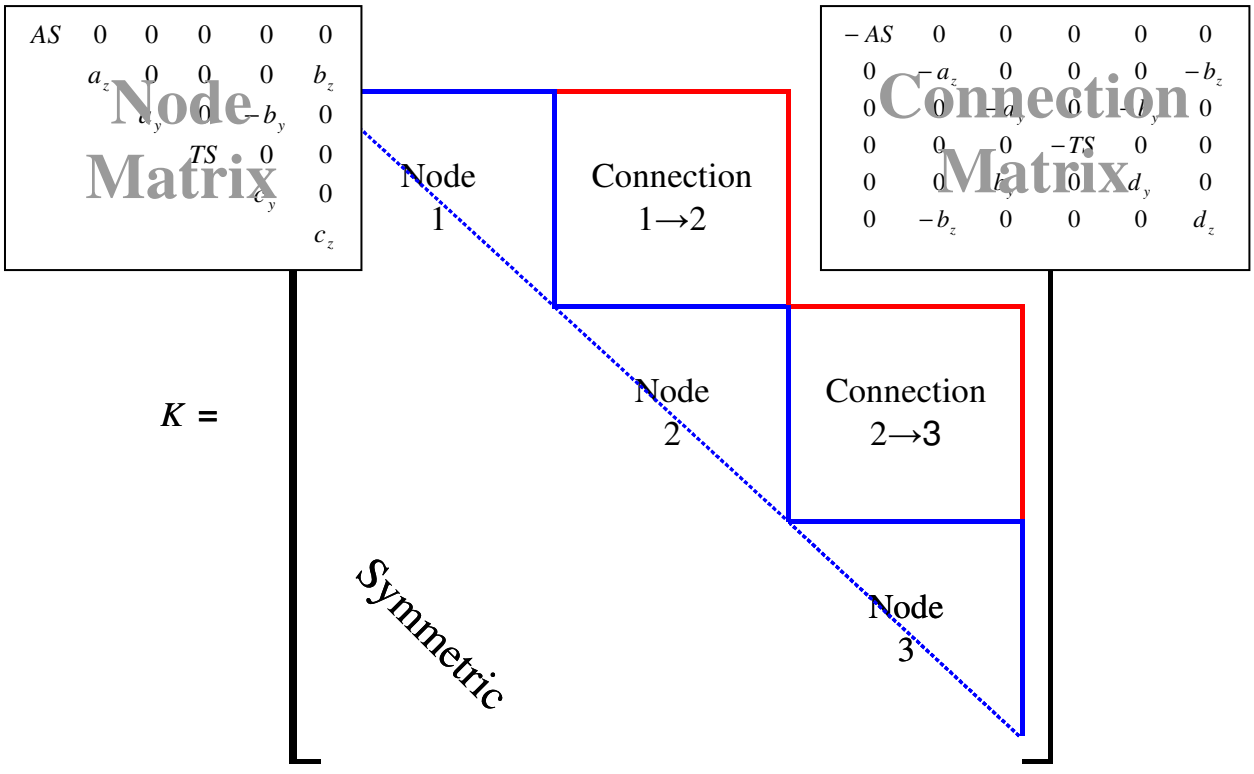


Figure 2.5: Sub-Categorization of Nodal and Connection Matrices

As stated before, the global stiffness matrix K for this system has a dimension of 18x18. When broken into 6x6 nodal submatrices, it's easy to see the distinction between each node and the associated connectivity, as shown in Figure 2.5. Note that the outermost nodal and connection matrices mirror the submatrices of Equation 2.41. The inner matrices, node 2 in Figure 2.5, will be different due to the connectivity within the component.

From the nodal point of view, the component stiffness matrix k_c can be divided into submatrices that have distinct sections dependent upon individual nodal characteristics, from which a symbolic representation can be established. We created a numbering system by representing each 6x6 nodal matrix by a single variable designated as N and its respective component node number. The connection 6x6 submatrix will be represented as C with a subscript indicating the nodes connected. We demonstrate the symbolic representation with the five node beam shown in Figure 2.6 and the component stiffness matrix k_c in Equation 2.42.

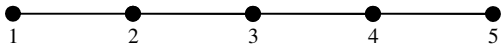


Figure 2.6: Five Node, Four Element Component

$$k_c = \begin{bmatrix} N_1 & C_{12} & 0 & 0 & 0 \\ & N_2 & C_{23} & 0 & 0 \\ & & N_3 & C_{34} & 0 \\ & & & N_4 & C_{45} \\ & & & & N_5 \end{bmatrix} \tag{2.42}$$

2.3 Nodal Placement

When generating a large FE model with multiple components, having the capability to readily determine which nodes are used to connect each component is ideal. Therefore, the appropriate placement of nodes within a component is essential. To discuss the nodal

placement, we will utilize the five-node beam shown in Figure 2.6, with the previously established notation for the symbolic representation of the matrices, as seen in Equation 2.42.

One attribute of the FE process is the ability to separate each element from the next. Because of the sub-structures we have defined, this capability can be taken one step further with the separation of the node and the connection matrices from the element structure. With this in mind, we separate the beam into three sections demonstrated in Figure 2.7; the left connection node, the body, and the right connection node. These separations are applied to the component stiffness matrix k_c as can be seen in Equation 2.43.

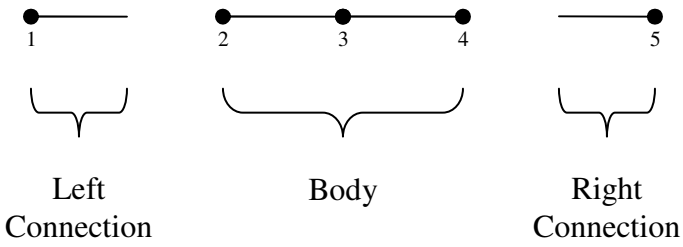


Figure 2.7: Segmented Five-Node, Four-Element Component

$$k_c = \left[\begin{array}{c:cccc} N_1 & C_{12} & 0 & 0 & 0 \\ \hdashline & N_2 & C_{23} & 0 & 0 \\ & & N_3 & C_{34} & 0 \\ & & & N_4 & C_{45} \\ \hdashline & & & & N_5 \end{array}\right] \tag{2.43}$$

With the simplified submatrix notation established, we can look at the location of the nodes within the matrix itself. Nodal placement within the matrix can be adjusted as long as the connection segment is placed properly. If a node is moved to the right of a connecting node, within the matrix only, the connection matrix is transposed. For instance, if we move node one within the matrix, not the beam itself, the connection matrix between nodes one and two is transposed (Equation 2.44). But, if a node is moved and remains to the right of the

node it is connected to, the connection matrix is not changed, demonstrated by nodes two and three.

$$k_c = \begin{bmatrix} N_2 & C_{23} & [C_{12}]^T & 0 & 0 \\ & N_3 & 0 & 0 & 0 \\ & & N_1 & C_{34} & 0 \\ & & & N_4 & C_{45} \\ & & & & N_5 \end{bmatrix} \tag{2.44}$$

From these movements, we can see that if the correct connection properties are maintained, the nodes within a given beam can be placed anywhere within the appropriate matrices. These practices can be utilized for any size or shape of beam.

Knowing that we can move the nodes within the matrix with little effort, we place the stiffness submatrix for the body before the first and last connecting nodes.

$$k_c = \left[\begin{array}{ccc:c:c} N_2 & C_{23} & 0 & (C_{12})^T & 0 \\ & N_3 & C_{34} & 0 & 0 \\ & & N_4 & 0 & C_{45} \\ \hdashline & & & N_1 & 0 \\ \hdashline & & & & N_5 \end{array}\right] \tag{2.45}$$

Within the five node example beam, the component node numbers do not hold significance other than to designate a location within the component FE model. Therefore, we re-number the nodes of the beam to coincide with our matrix movement and place the body before the connectors (Figure 2.8), providing the same matrix as above, but with sequential node numbering.

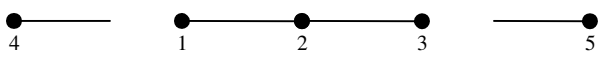


Figure 2.8: Segmented, Re-Numbered, Five-Node, Four-Element Component

$$k_c = \left[\begin{array}{ccc:c:c} N_1 & C_{12} & 0 & C_{14} & 0 \\ & N_2 & C_{23} & 0 & 0 \\ & & N_3 & 0 & C_{35} \\ \hdashline & & & N_4 & 0 \\ \hdashline & & & & N_5 \end{array}\right] \tag{2.46}$$

The transpose of the connection between nodes one and four is removed due to the implied direction of the connection. After re-numbering the local nodes, the connection is from node one to node four, which inherently transposes the original connection, which maintained the direction of node four to node one.

The configuration detailed above is an advantageous node order for components within a FE model. By placing the body of the component before the connecting nodes within the stiffness matrix, the connecting nodes are always easily accessible due to the position they occupy, regardless of the number of nodes of the component. For instance, if the beam we used in the previous example is increased in resolution to fifty nodes (Figure 2.9), the stiffness matrix in Equation 2.47 increases in size, but the connection nodes remain in the last diagonal positions. The bolded zero, **0**, represents the upper triangular remainder of the body matrix, which are all zeros.

Figure 2.9: Segmented, Re-Numbered, Fifty Node, Forty-Nine Element Component

$$k_c = \left[\begin{array}{ccc:c:c} N_1 & C_{1\vert\cdots} & & \mathbf{0} & C_{1\vert 49} & 0 \\ & \ddots & \ddots & & 0 & \vdots \\ & & \ddots & C_{\cdots\vert 48} & \vdots & 0 \\ & & & N_{48} & 0 & C_{48\vert 50} \\ \hdashline & & & & N_{49} & 0 \\ \hdashline & & & & & N_{50} \end{array}\right] \tag{2.47}$$

Separating the connection nodes from the body is advantageous when we assemble a model with multiple connected components. A methodology is used to define the storage requirements for the component. The storage of the segregated matrix will require separation into five segments: body, connector 1 (Cnctr1), connection 1 (Cnx1), connector 2 (Cnctr2), and connection 2 (Cnx2).

$$k_{Body} = \begin{bmatrix} N_1 & C_{1|\cdots} & & \mathbf{0} \\ & \ddots & \ddots & \\ & & \ddots & C_{\cdots|48} \\ & & & N_{48} \end{bmatrix} \tag{2.48}$$

$$k_{Cnctr1} = \left[N_{49} \right] \tag{2.49}$$

$$k_{Cnx1} = \left[C_{1|49} \right] \tag{2.50}$$

$$k_{Cnctr2} = \left[N_{50} \right] \tag{2.51}$$

$$k_{Cnx2} = \left[C_{48|50} \right] \tag{2.52}$$

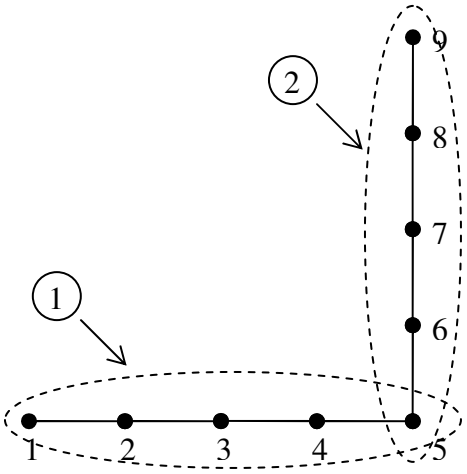
The re-combined component stiffness matrix is shown in Equation 2.53.

$$k_c = \begin{bmatrix} \begin{bmatrix} & & \\ & k_{Body} & \\ & & \end{bmatrix} & \begin{matrix} [k_{Cnx1}] \\ \vdots \\ 0 \\ [k_{Cnctr1}] \end{matrix} & \begin{matrix} 0 \\ \vdots \\ [k_{Cnx2}] \\ 0 \end{matrix} \\ & & [k_{Cnctr2}] \end{bmatrix} \tag{2.53}$$

By inspection, it is seen that the segmented and un-segmented matrices are equivalent.

2.4 Construction of Global Stiffness Matrix

Construction of an entire FE model utilizing the methods previously described significantly reduces the complication and ambiguity of the standard global *K*. The traditional assembly method for a complex system of components yields a global stiffness matrix *K* that has no clear indications of the location of component connectivity. For example, we look at the structure in Figure 2.10 (a) and the associated *K* matrix (b).



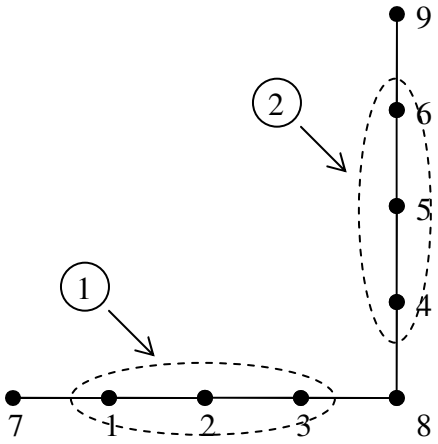
(a)

$$K = \begin{bmatrix} N_1 & C_{1|2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ & N_2 & C_{2|3} & 0 & 0 & 0 & 0 & 0 & 0 \\ & & N_3 & C_{3|4} & 0 & 0 & 0 & 0 & 0 \\ & & & N_4 & C_{4|5} & 0 & 0 & 0 & 0 \\ & & & & N_5 & C_{5|6} & 0 & 0 & 0 \\ & & & & & N_6 & C_{6|7} & 0 & 0 \\ & & & & & & N_7 & C_{7|8} & 0 \\ & & & & & & & N_8 & C_{8|9} \\ & & & & & & & & N_9 \end{bmatrix}$$

(b)

Figure 2.10: (a) Graphical Representation and (b) Global Stiffness Matrix of Two Component Structure without Nodal Segregation

Examination of the K matrix without the graphical representation reveals no information as to where the connecting node is located. When using this nodal segregation approach, the separation of connecting nodes from the body nodes, the connecting node is obvious, demonstrated in Figure 2.10 (b). Due to the guidelines established in section 2.3, the local component numbering scheme can be modified in any matter as long as the connectivity characteristics are maintained. Therefore, we are able to renumber the local component nodes to place the bodies of the components at the top left of the global stiffness matrix K. The numbering modifications also apply to the connecting nodes, and are numbered last for ideal placement within K.



(a)

$$K = \begin{bmatrix} \begin{bmatrix} N_1 & C_{1|2} & 0 \\ & N_2 & C_{2|3} \\ & & N_3 \end{bmatrix} & & & C_{1|7} & 0 & 0 \\ & 0 & & 0 & 0 & 0 \\ & & & 0 & C_{3|8} & 0 \\ & \begin{bmatrix} N_4 & C_{4|5} & 0 \\ & N_5 & C_{5|6} \\ & & N_6 \end{bmatrix} & & 0 & C_{4|8} & 0 \\ & & & 0 & 0 & 0 \\ & & & 0 & 0 & C_{6|8} \\ & & & [N_7] & 0 & 0 \\ & & & & [N_8] & 0 \\ & & & & & [N_9] \end{bmatrix}$$

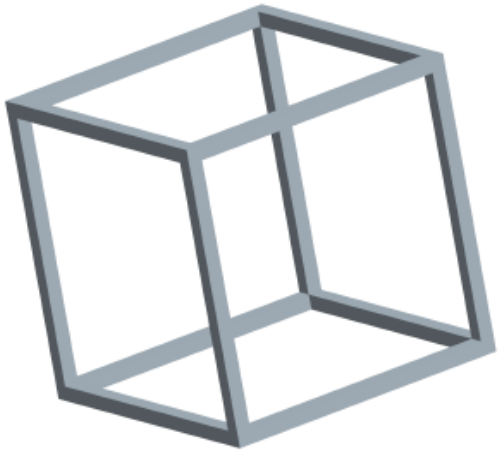
(b)

Figure 2.11: (a) Graphical Representation and (b) Global Stiffness Matrix of Two Component Structure Utilizing Nodal Segregation

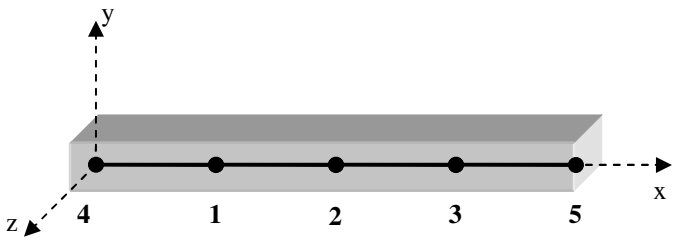
By examining the global stiffness matrix *K*, shown in Figure 2.11, we can demonstrate the benefits of this nodal segregation approach. The connectivity of the two elements is readily available; from Figure 2.11 (b), we can easily see that node eight is the connecting node between components one and two. We can also determine the end points if further uses of those are needed. Independent of the size or number of nodes a component has, if this nodal segregation approach is used, the defining connectivity information will be accessible.

2.5 Model Construction

For realistic FE model generation, a more complex structure must be examined. Figure 2.12 (a) shows a cube structure created from identical beams on each side. This avails the development of a single component model for the beam, and the reuse of it twelve times.



(a)



(b)

Figure 2.12: (a) Cube Structure and (b) Associated Component

The FE model for this component is a repeat of the beam utilized in the previous sections. The nodal segregation is maintained within the component's k_c matrix at this point. The example is a five node component, and therefore, the body matrix, k_{Body}, dimension is 18x18 while the connector node, k_{Cnctr}, and connection node, k_{Cnx}, matrices are 6x6. For each component, the component stiffness matrix k_c maintains a 30x30 dimension. Equation 2.54 depicts the k_c matrix for each component within the cube structure.

$$k_{cj} = \begin{bmatrix} \begin{bmatrix} & & \\ & k_{Body} & \\ & & \end{bmatrix} & \begin{matrix} [k_{Cnx1}] \\ \vdots \\ 0 \\ [k_{Cnctr1}] \end{matrix} & \begin{matrix} 0 \\ \vdots \\ [k_{Cnx2}] \\ 0 \end{matrix} \\ & & [k_{Cnctr2}] \end{bmatrix} \quad \text{where} \quad j = 1 \cdots 12 \tag{2.54}$$

Before the individual component FE models are assembled to form the global stiffness matrix K, each component stiffness matrix must be transformed from local to global coordinates. As in Equation 2.37, the transformation of the component stiffness matrix k_c will be dependent on the cosines between two vectors; the component unit vector and a global axis. The transformation can be equated by using a Rodrigues' angle rotation formula. Rodrigues' formula is similar to the Euler angle transform matrix but uses an axis vector and angle to calculate the transformation matrix where the Euler method uses three axis angles. Either method may be applied in this situation, but Rodrigues' use of vectors is more convenient for these models due to the availability of a designated positioning unit vector within each component FE model.

To demonstrate the use of the Rodrigues' formula, we will rotate the beam component in Equation 2.48 to align with the global y-axis. The unit vector for the component, V_u, is (1,0,0) because it is oriented along the x-axis, and the global placement vector, V_{GP}, is (0,1,0). Figure 2.13 shows the specified rotation.

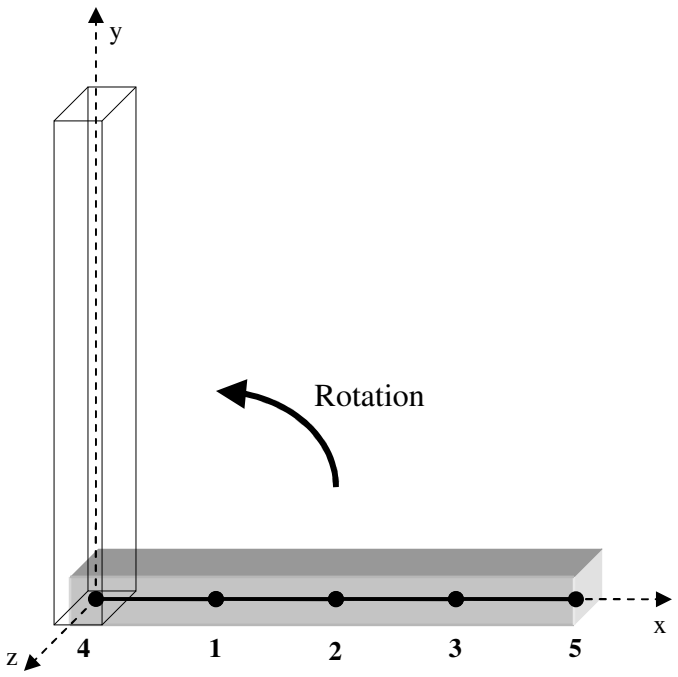


Figure 2.13: Rotation of Beam Component from X-Axis to Y-Axis.

$$
\begin{aligned}
V_U &= \begin{pmatrix} 1 & 0 & 0 \end{pmatrix} \\
V_{GP} &= \begin{pmatrix} 0 & 1 & 0 \end{pmatrix} \\
V_{Cross} &= V_U \times V_{GP} \\
V_{Dot} &= V_U \bullet V_{GP} \\
N &= \left\| V_{Cross} \right\|
\end{aligned}
\tag{2.55}
$$

If V_u is equal to V_{GP}, the magnitude of the cross product, N, is equal to zero and the rotation is then only dependent on the dot product of the two vectors (V_{Dot}). If V_{Dot} is equal to one, the matrix is not rotated and is placed as is. If V_{Dot} is equal to negative one, the axis vector (A) is defined as (0,-1,0) Otherwise, the following step is taken for calculation of A.

$$
\begin{aligned}
A &= \frac{V_{Cross}}{N} \\
A &= \begin{bmatrix} a_1 & a_2 & a_3 \end{bmatrix}
\end{aligned}
\tag{2.56}
$$

Despite the method for calculating *A*, the cosine and sine of the axis angle are computed for use in the Rodrigues' transformation formula *R*, developed in Equation 2.57.

$$R = \begin{bmatrix} \left(c + A_1{}^2 \cdot (1-c)\right) & \left(A_1 \cdot A_2 \cdot (1-c) - s \cdot A_3\right) & \left(A_1 \cdot A_3 \cdot (1-c) + s \cdot A_2\right) \\ \left(A_1 \cdot A_2 \cdot (1-c) + s \cdot A_3\right) & \left(c + A_2{}^2 \cdot (1-c)\right) & \left(A_2 \cdot A_3 \cdot (1-c) - s \cdot A_1\right) \\ \left(A_1 \cdot A_3 \cdot (1-c) - s \cdot A_2\right) & \left(A_2 \cdot A_3 \cdot (1-c) + s \cdot A_1\right) & \left(c + A_3{}^2 \cdot (1-c)\right) \end{bmatrix} \tag{2.57}$$

where,

$$\begin{aligned} \theta &= \cos^{-1}\left(\frac{V_{Dot}}{\left(\|V_u\| \cdot \|V_{GP}\|\right)}\right) \\ c &= \cos(\theta) \\ s &= \sin(\theta) \end{aligned} \tag{2.58}$$

Equation 2.34 defines the directional cosine matrix as λ. The Rodrigues derivation is the same matrix, just derived using different inputs. Therefore, we substitute *R* for *λ* within the diagonal transformation matrix *L* and get Equation 2.59.

$$L = \begin{bmatrix} R & & 0 \\ & \ddots & \\ 0 & & R \end{bmatrix} \tag{2.59}$$

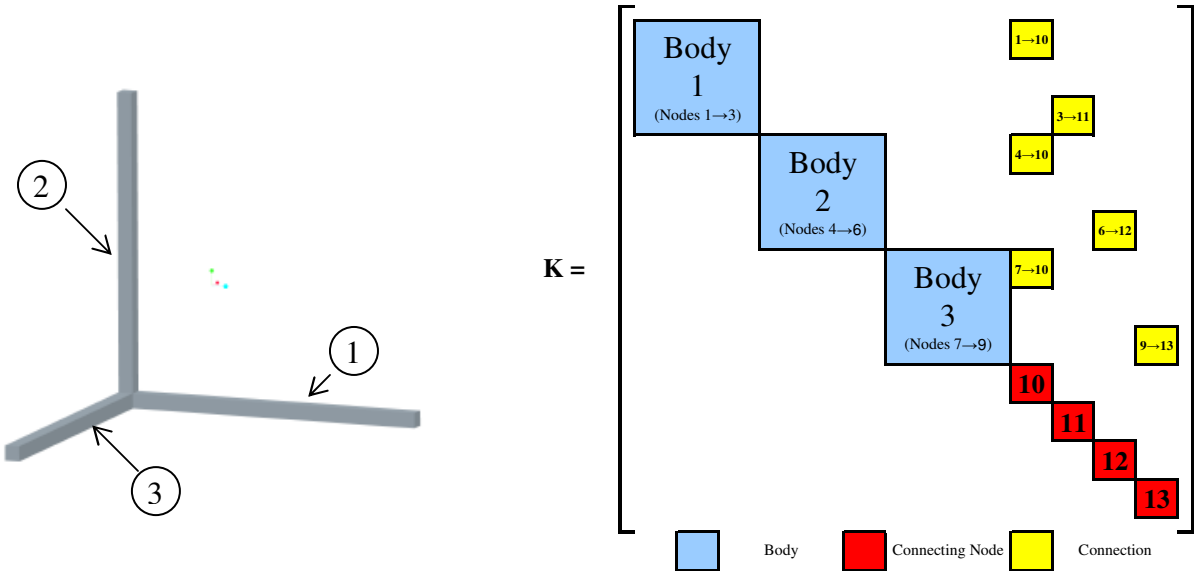
To perform the transformation on the component's *k'* matrix, the standard matrix multiplication method is used. The transformed component matrix will be designated as k_c.

$$k_c' = L^T k_c' L \tag{2.60}$$

For proper assembly of our example, the cube structure, this transformation will need to occur for all twelve components. For four beams, the transformation will not change the matrix, as the placement vector is in the x-direction, matching that of the beam model.

With the utilization of Rogrigues' formula for matrix transformation in place, generation of the global model is initiated. We begin by placing the first component in the x-direction.

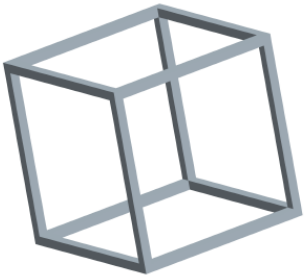
As previously mentioned, there is no modification necessary because the unit vector is in the same direction as the global placement vector. Therefore, the global K at this point is shown in Equation 2.53. Placement of the next two components, one in the y-direction (V_{GP} = [0 1 0]), and one in the z-direction (V_{GP} = [0 0 1]), will make the dimension of K equal 78x78, demonstrated in Figure 2.14. The component nodes are re-numbered to the global node numbering scheme each time maintaining the connection points at the bottom right corner of the matrix.



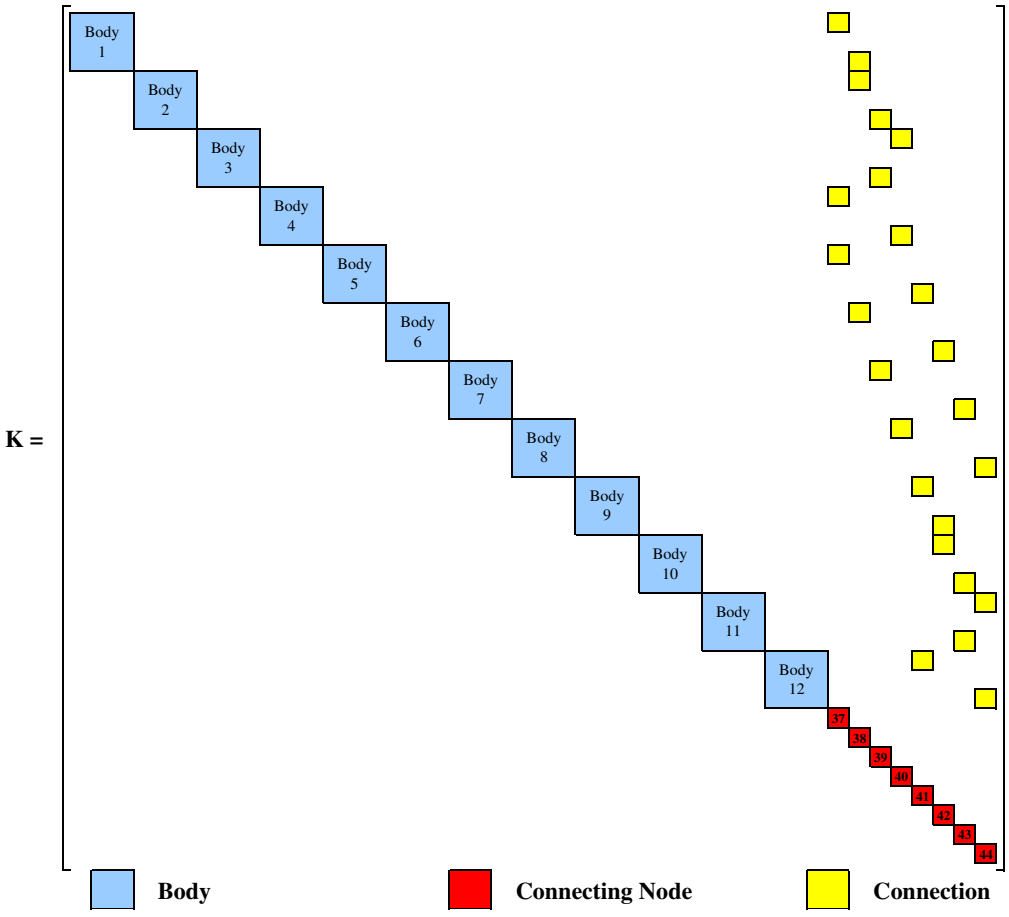
(a) (b)

Figure 2.14: (a) Three Beam assembled Structure and (b) Associated Global Stiffness Matrix K

By utilizing these component submatrix generation tactics on the remaining nine components within the cube structure, we develop the complete global stiffness matrix K. The 264x264 matrix is shown in Figure 2.15.



(a)



(b)

Figure 2.15: (a) Assembled Cubic Structure and (b) Associated Global Stiffness Matrix K

2.6 Matrix Storage Techniques

The next step within the nodal segregation concept is the storage of the component submatrices. The body portion of the component stiffness matrix k_c can be stored separate from the connector node and connection matrices. If the five node beam used previously (Figure 2.12) is segregated for storage, a total of three separate submatrices can be derived; the body, connector node, and connection matrices.

Storing the component submatrices separately works well for a system whose design is fixed with no potential changes. But, what if we want to refine the mesh, or increase the resolution, of a component FE model without modifying the entire global K?

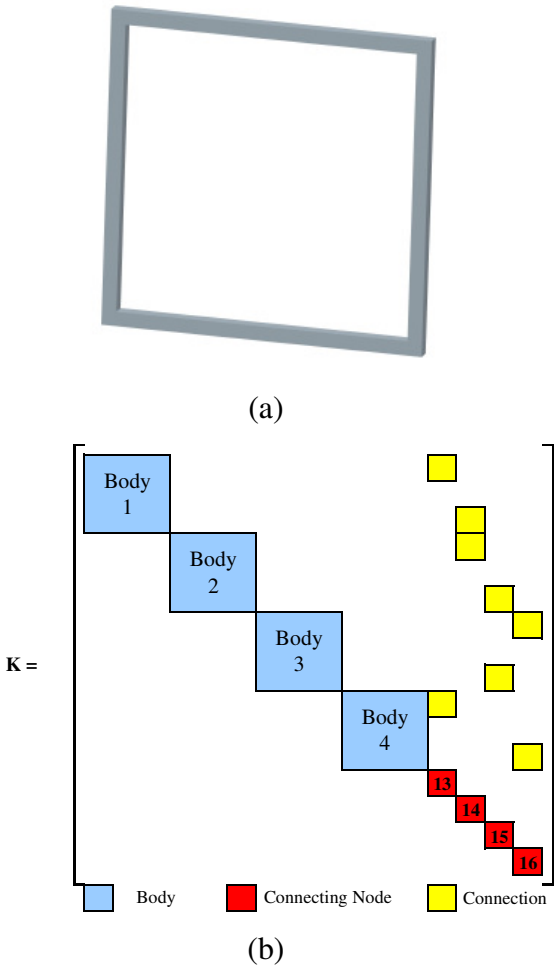
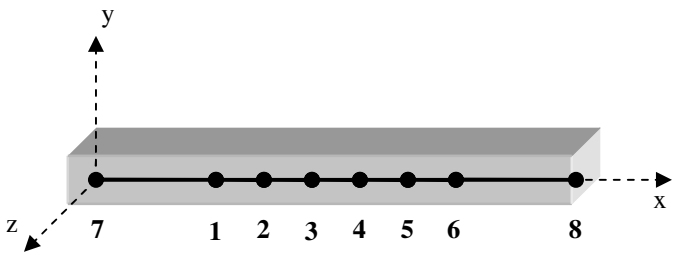
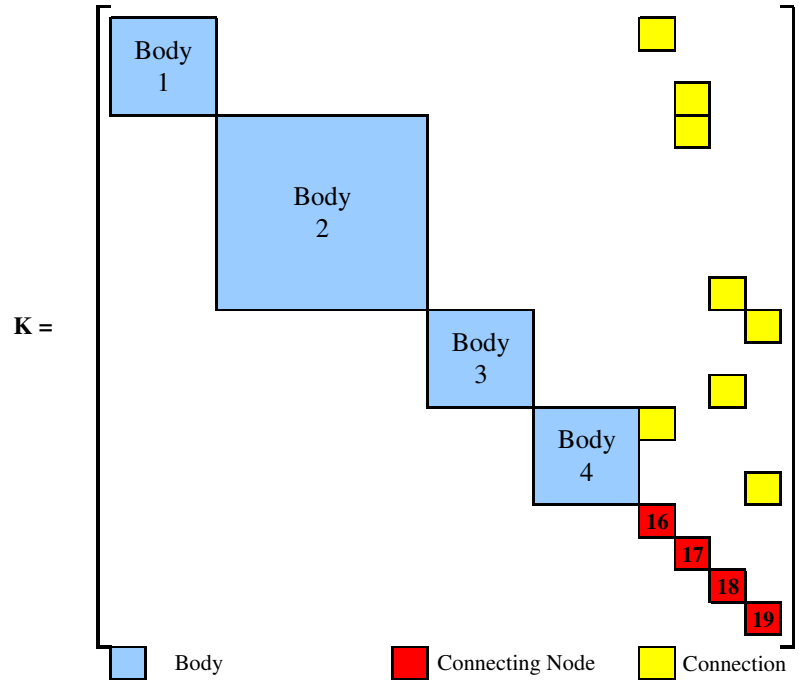


Figure 2.16: (a) Assembled Square Structure and (b) Associated Low-Resolution K Matrix

For instance, if we take the square beam structure in Figure 2.16 (simplified from the previously discussed cube), but want to redefine the mesh on one beam, what effect does this have on our global *K*? If the model is modified using a straight forward, evenly spaced node distribution, all three of the pertinent beam matrices (Body, Connecting Node, and Connection) are changed; necessitating a complete restructure of the global *K*. But, if we define a component development rule that requires the placement of the connecting nodes at the end of the components and the distance between those nodes and the body nodes must remain constant, the only change to the global stiffness matrix *K* is the dimension of the body matrix. For example, the number of nodes within the body of beam two has been doubled (Figure 2.17(a)) which increases the DOF from 18 to 36. The placement of the connecting node and connection matrices do not change with respect to the body matrix for beam number two (Figure 2.17(b)).



(a)



(b)

Figure 2.17: (a) High Resolution Beam Nodal Placement and (b) Structure *K* Matrix

Note that even though the size of the body matrix doubled, the placement of the connection matrix remains constant at the first and last nodes. This would be consistent with any size body matrix, as long as the connection remains constant.

2.7 Connection Interface Rules

To maintain a consistent interface for each component within a model, certain guidelines have been established and must be strictly followed when utilizing this nodal segregation approach. The following sum up the development criteria into two dominant rules.

1. For any number of nodes that the component has, the distance between any connection node and the closest body node must maintain a fixed distance.
2. When the mesh is numbered, all connection nodes are numbered last.

When these rules are maintained, the body and connection matrices can be moved within the global matrix independent on the size of the body matrix. This methodology also allows us to vary the number of components in a model and the resolution of each component to produce adequate results for a system.

3 Application of Nodal Segregation within Plug-and-Play Structures

The methodology behind the nodal segregation presented in this thesis is directly applicable to the modeling of the Plug-and-Play (PnP) structures utilized within the Operational Responsive Space (ORS) effort. By utilizing the FE model construction technique and strictly following the development rules, a complete model for a finalized configuration can be assembled and analyzed within the ORS timeframe.

3.1 ORS Program Requirements for Successful FE Analysis

Due to the non-traditional timeline that ORS follows, described in Chapter One, each component must reach technological maturation prior to any inclusion in satellite design, typically a one to three year process. The key to the success of the ORS program is the adherence to the provided guidance for the development and interoperability of the component's interface. When a mission is defined and it requires the componentry to be assembled, the maturity level and consistent interface ensure that the assembly is successful.

The paradigm that is used for hardware development is applied to the creation of FE models of the entire structure. The modeling of individual components would follow the same acquisition process as the hardware development. The creation of each component FE model should be an iterative process throughout the hardware design cycle, validating the computed response against actual harmonic excitation of the hardware to achieve accurate and representative FE models.

Similar to the hardware design, the component FE models will follow an established set of parameters that govern the interface of each model. Specifically, the number of connecting nodes and the distance between the connecting node and the closest neighboring body node must remain constant for each component. By maintaining an interface convention within each model, the system placement of each component can be achieved in a fraction of the time required by traditional methods.

3.2 Component Model Development

If a FE model is to be generated for the entire system, each individual component's model must accurately reflect the characteristics of the component. Without an appropriate validation process for each component model, the analysis of the model has potential to provide inaccurate results. The usage of such models within an assembled structure would invalidate any results achieved during system analysis.

The advantage of the approach presented here is that several component models can be created each with varying mesh refinement or resolution dependant upon specific load or vibrational concerns. In addition, a low resolution, or low nodal count, model could be generated for initial analysis. The generation of the specified series of models for each component will allow for the complete satellite model to be refined at the component level while maintaining a fully validated model.

Similar to the example problem presented in Chapter 2, a model's mass and stiffness matrices would be broken into submatrices according to the previously developed procedures. Each component's body, connector node and connection submatrices would be stored separately, but, due to the requirement that the connection node maintain the same distance to the closest body node, only the body matrices would vary. The connector node and associated connectivity matrices would remain unchanged for all component models.

3.3 Resolution Modifications within the Compiled FE Model

After FE analysis of any system is completed, interpretation of the analysis results provide specific areas that will need further examination. Typically, modifying the system model to incorporate the higher resolution necessary is a time consuming process. Specifically, the refined portion of the mesh must be inserted manually to ensure all model connectivity remains intact. But, if the governing mesh interface design parameters are maintained, the swapping of the different resolution models can be effortless.

The mesh refinement within Section 3.5 demonstrates that even though the resolution of the body of the beam is doubled, the attachment of the connecting nodes remains constant on the first and last node of the body. While the models within the ORS program will be

significantly more complex, the basic theory still applies. Independent of the number of connection points within a given model, the explicit amount and their specified locations remain constant, allowing for the immediate replacement of models when specified. The resolution of any given component within a compiled model can therefore be changed at will, as long as the required model is produced and validated prior to system assembly. By utilizing the nodal segregation approach within design and assembly of the component FE models, the time savings received ensures this approach is favorable with respect to the ORS program.

3.4 Plug-and-Play Sample Structure

To demonstrate the Plug-and-Play (PnP) concept and how the approach developed in this research applies to a structural model, we designed and produced a small, scaled down sample satellite structure, called SimpleSat. SimpleSat was created to demonstrate how the main structural components of a satellite can be assembled by utilizing pre-fabricated structural components. In mimicking the established guidelines for the ORS program, we have produced a selection of parts that can be utilized for construction in a variety of configurations. SimpleSat is for demonstration purposes only, therefore the configuration options are limited.

3.4.1 SimpleSat Design

All components produced within SimpleSat were designed using Pro/Engineer, a solid modeling application produced by Parametric Technologies Corporation. All of the parts were made of either 6061-T6 or 2024-T4 aluminum.

Individual Components

A total of eight separate machined components were created in the development of SimpleSat (see Figure 3.1). The body structural beams, measuring four, six, and eight inches, were constructed from half inch round 2024 stock. Each beam has a #8X32 threaded hole at each end for attachment. The representative solar panel measures four inches by

seven inches and is constructed from 0.05 inch thick aluminum sheet. The solar cross member is one 0.25 inches thick and measures six inches from each hole, measured horizontally and vertically. Both the panel and cross member were cut from aluminum sheet using a high pressure water cutting machine. The attachment collar was produced from 0.375 inch round stock and turned on a machine lathe to achieve the shoulder feature and drilled to make it hollow.

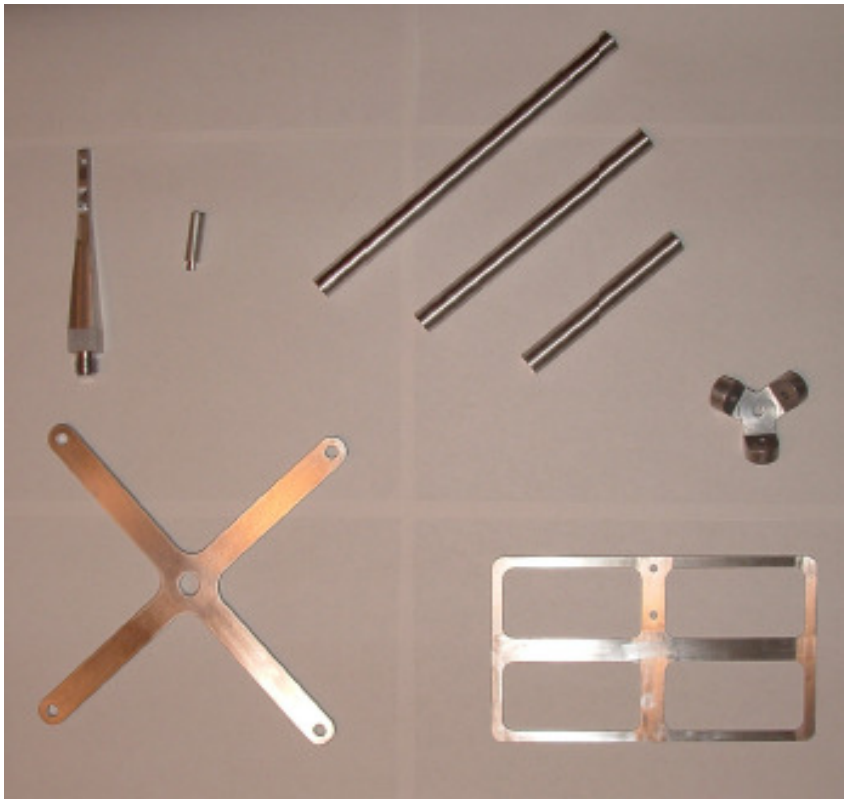
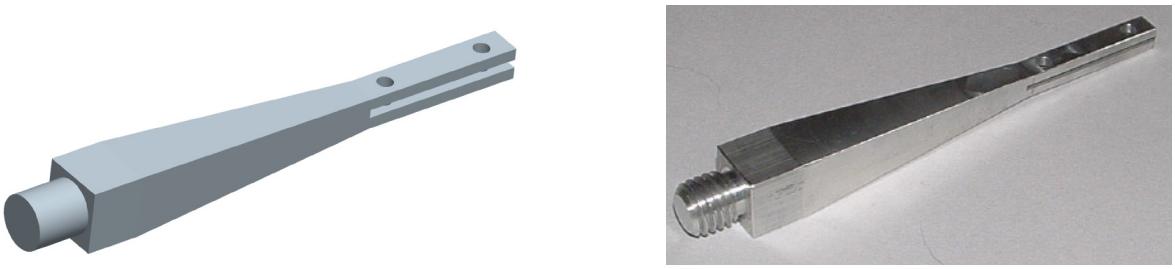


Figure 3.1: Machined SimpleSat Components

Two components within the system were machined using a standard machine mill and were time and manpower intensive. The first, the solar panel support, was created from 0.625 square aluminum stock. To achieve the desired strength and maintain a lightweight component, a linear taper was machined from the connection shoulder to the solar panel connecting fork. The part was then turned using the machine lathe to produce the attachment threads. The solid model and final product can be seen in Figures 3.2 (a) and (b) respectively.



(a) (b)

Figure 3.2: (a) Solar Panel Support Pro/Engineer Model and (b) Machined Component

(a)

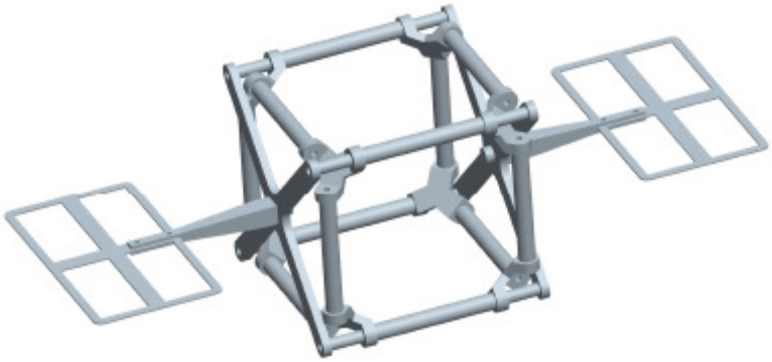
(b)

Figure 3.3: (a) Corner Unit Pro/Engineer Model and (b) Machined Component

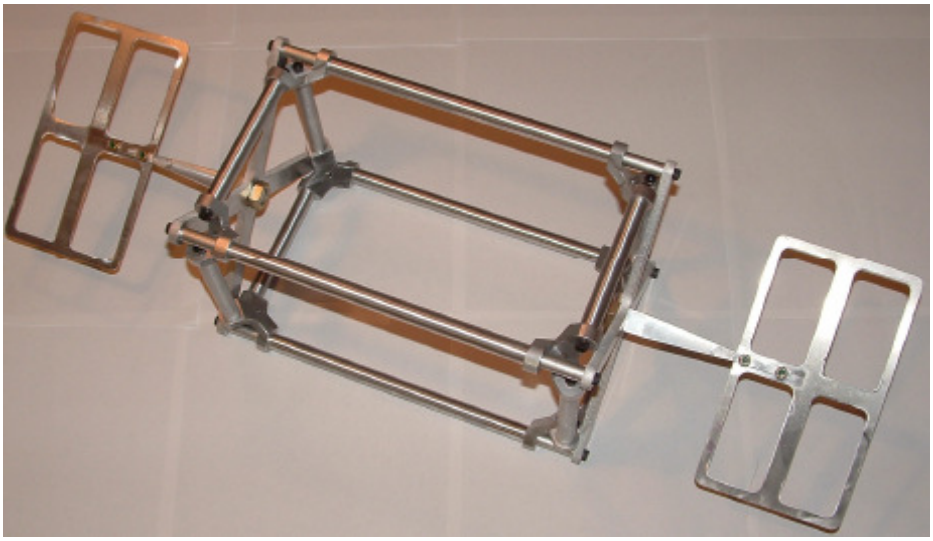
The structural connecting component, the corner unit, is the most geometrically complicated piece within the entire SimpleSat structure; designed to maintain the cubic shape of the satellite body while retaining the ability to connect the solar panel assembly and be disassembled easily. Many different configurations were examined prior to production, but all had conflicts within the assembly. Figure 3.3 (a) depicts the Pro/Engineer solid model and Figure 3.3 (b) shows final machined component. The corner component was cut from a 0.625 inch thick sheet of aluminum using the high pressure water jet cutting machine. The unit was then drilled through the center; this hole was not modeled but was required for milling. A custom indexing jig was created to mill the angled attachment points. The machining of this piece was complicated by the antiquated machinery used and resulted in three to four hours of machining for each unit constructed. By utilizing a modern five-axis Computer Numerical Code (CNC) machine mill, the process time could be reduced by as much as seventy-five percent and the setup would be a one-time pre-machining event.

3.4.1.1 Assembled Structure

By utilizing the eight components described previously, a complete structure can be assembled. Due to design limitations, the only Plug-and-Play components that can change the configuration of SimpleSat are the structural beams. The longitudal beams within the model have the ability of being exchanged within the assembled structure to change the model length. This is demonstrated in Figure 3.4 as the Pro/Engineer model (Figure 3.4 (a)) is constructed with four inch beams and the machined assembly pictured (Figure 3.4 (b)) utilizes the eight inch beams in the longitudal direction. The SimpleSat model is assembled using standard size eight socket-head screws with thirty-two threads per inch (#8X32).



(a)



(b)

Figure 3.4: (a) SimpleSat Pro/Engineer Model and (b) Assembled SimpleSat Hardware

3.4.2 Computer Modeling and Simulation

As with the modeling of the structure for physical production, SimpleSat was modeled for analysis and system simulation. In order to demonstrate the basic application of the nodal separation approach described in Chapter 2, multiple FE models were produced for each individual component, varying resolution in each. The models were stored on disk, ready for assembly within the global model. The assembly process starts with the development of a

component connectivity plan, called the Model Tree. From the Model Tree, the code developed in this research assembles the complete FE model from the components selected and initiates the FE analysis.

For all FE model construction and analysis, the Structural Dynamic Toolbox (SDT) was utilized. SDT is a MATLAB® toolbox created and distributed by Etienne Balmes, and is used throughout the world for complex FE modeling and analysis. SDT develops the FE model using proprietary code and takes advantage of the MATLAB® calculation engine for solution computation. SDT is an ideal product for this research because we were able to manipulate the mass and stiffness matrices at the nodal level.

3.4.2.1 Finite Element Model Development

The production of each FE model began with the definition of required connection nodes and applicable distances. For consistency, the connection length for all connecting nodes within the structure is set at 0.1 inch. All nodes within the body of a component are spaced evenly between the remaining distances. For instance, the low resolution, four-inch beam has a total of ten nodes; the first and last node spacing is set at 0.1 inch from the nearest body nodes, while the remaining node spaces are equally spaced at 0.543 inches each.

The resolution for each part is of interest as well. Three specific parts were of major concern: the solar panel, attachment collar and structural beams. The structural beams were modeled with three resolutions for each length. The nine instances of the beams consist of ten, one hundred, and one thousand nodes per the component lengths of four, six, and eight inches. Figure 3.5 depicts the SDT model of the four inch beam modeled with ten nodes. As can be seen, the body of the beam is represented with nodes one thru eight and the connection nodes are nodes nine and ten. The constant connection length on 0.1 inch is also shown. The attachment collar is modeled very similarly to the structural beams with the three separate resolutions.

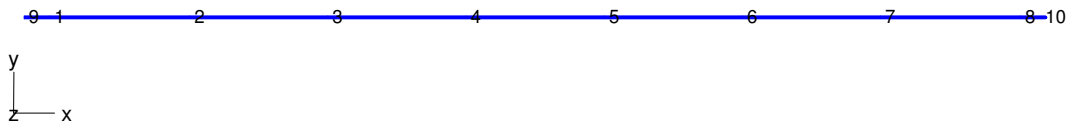
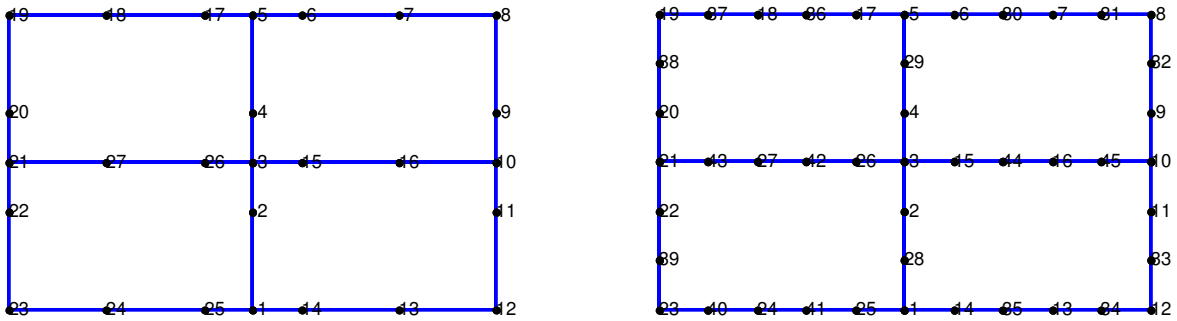


Figure 3.5: SDT model of Low Resolution Four Inch Beam

The modeling of the solar panel is delineated by two separate resolutions. The size is constant; therefore the resolution differs only in the distance between nodes within the body. The lower resolution model incorporates a total of twenty-seven nodes while the higher resolution model uses forty five. Again, the distance between the connecting node and the body nodes is 0.1 inch. Figure 3.6 shows the resolution differences between instances of the solar panel.



(a) (b)

Figure 3.6: SDT model of (a) the Low Resolution and (b) the High Resolution Solar Panel

A naming convention for each model has been established based on the component name, the change in length, and the resolution of the model i.e. *Component.Length.Resolution.* Each developed component follows the naming convention to the extent that if the model does not utilize a field, the field is eliminated. For instance, the corner component does not require a length change so the length field is eliminated.

Table 3.1: SimpleSat FE Model stored files

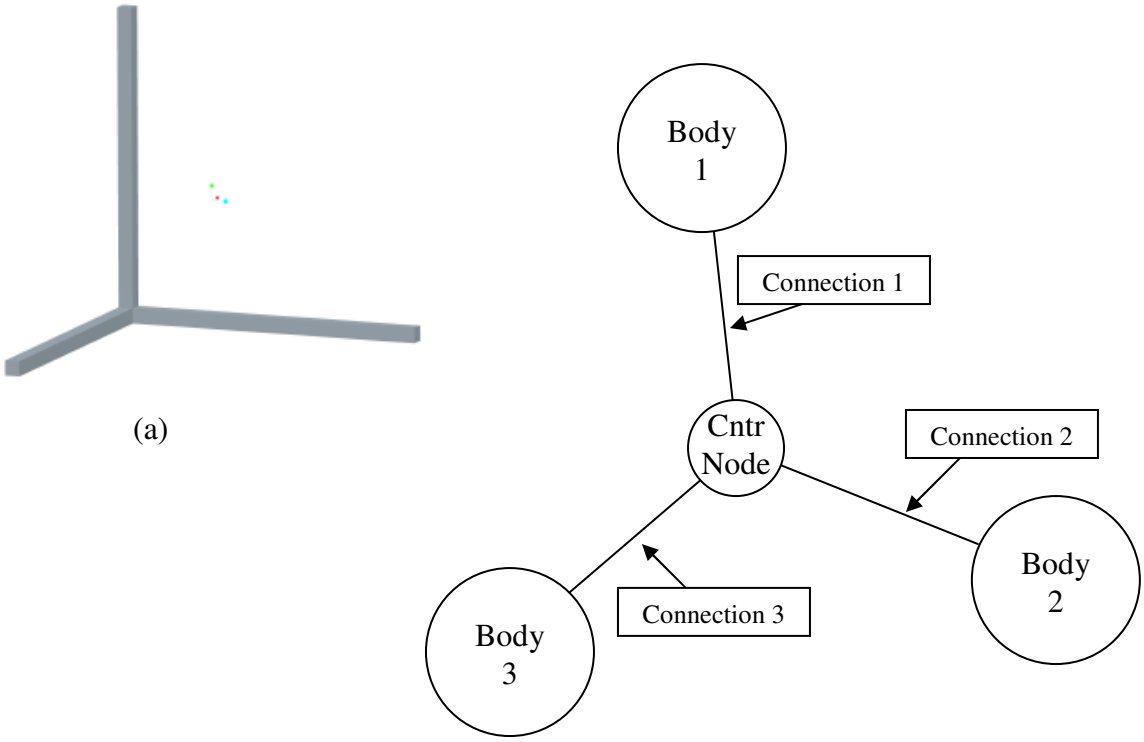
Component	Length (inches)	Resolution	File Extension	# of Nodes	Matrix Size
Beam	Four,	Low	.M	8	48 X 48
			.K		48 X 48
			.Nodes		10 X 7
			.Elt		10 X 6
	Six,	Med	.M	98	588 X 588
			.K		588 X 588
			.Nodes		100 X 7
			.Elt		100 X 6
	and Eight	High	.M	998	5988 X 5988
			.K		5988 X 5988
			.Nodes		1000 X 7
			.Elt		1000 X 6
Collar		Low	.M	8	48 X 48
			.K		48 X 48
			.Nodes		10 X 7
			.Elt		10 X 6
		Med	.M	98	588 X 588
			.K		588 X 588
			.Nodes		100 X 7
			.Elt		100 X 6
Solar Panel		Low	.M	27	162 X 162
			.K		162 X 162
			.Nodes		27 X 7
			.Elt		31 X 6
		Med	.M	45	270 X 270
			.K		270 X 270
			.Nodes		45 X 7
			.Elt		49 X 6
Corner			.M	12	72 X 72
			.K		72 X 72
			.Nodes		12 X 7
			.Elt		16 X 6
Solar Cross Member			.M	17	78 X 78
			.K		78 X 78
			.Nodes		17 X 7
			.Elt		17 X 6
Solar Support			.M	11	66 X 66
			.K		66 X 66
			.Nodes		11 X 7
			.Elt		11 X 6

Each FE model is stored electronically in two formats, graphical and analytical. SDT requires two model specific files for graphical analysis, the .Node, which gives the location of each node and the .Elt, which provides the element connectivity. The analytical file requirements are for two model specific files as well. The mass matrix for the component model is contained within the .M file and the .K file houses the global stiffness matrix. All component global mass and stiffness matrices are stored using the MATLAB® sparse storage format, storing only the non-zero terms, to minimize storage space and processing time. Table 3.1 provides a list of all of the stored FE models and their associated information for SimpleSat. Note that all stored components only contain the nodes for the body.

3.4.2.2 Model Tree

The assembly of the SimpleSat FE model begins with the definition of the Model Tree. The connectivity between each component and the physical placement of the component is all stored within the Model Tree file, and can be depicted graphically as a connectivity diagram. Figure 3.7 (b) demonstrates the connectivity for the three beam example shown in Figure 3.7 (a). The three bodies within the model connect with each other through the one connector node.

While the diagram depicted in Figure 3.7 is a good visual representation of the model connectivity, the Model Tree file is significantly more complex. The physical placement within the model is as important as the connectivity for the successful assembly of the global *M* and *K* matrices. The Model Tree contains three placement fields that are key to proper placement of the model: placement point, position vector, and rotation. As described in Chapter 2, each FE model was constructed with a reference unit vector for placement. The placement point dictates the spot within the graphical model where the reference vector initiates. The position vector provides the direction in which the reference vector should point. If a rotation about the positioned vector is required, that angle is entered in the rotation field.



(a)

(b)

Figure 3.7: (a) Three Beam Object and (b) the Associated Connectivity Tree

Table 3.2: Model Tree File Entries for Connecting and Non-Connecting Members

Model Tree Member Information				
Connecting Member			Non-Connecting Member	
Type		'Corner'	Type	'Beam.Four.Low'
Placement Point		[0 0 0]	Placement Point	[1 0 0]
Placement Vector		[1 0 0]	Placement Vector	[1 0 0]
Rotation		[0]	Rotation	[0]
Connecting Member		1	Connecting Member	0
Connection (1)	Member	1		
	Type	'Pos'		
Connection (2)	Member	4		
	Type	'Pos'		
Connection (3)	Member	5		
	Type	'Pos'		

To facilitate in model construction, each component has been given a designator of connecting member or non-connecting member. The purpose of the designation is to determine which member the connecting node should be built from. In the theoretical development of the models, assigning a member as connecting or non-connecting is not applicable. However, for real world model development, the distinction is made to dictate the correct connection with the appropriate body. In Table 3.2, the field "Connecting Member" determines whether a member is a connecting member (1) or a non-connecting member (0).

The connecting members within the Model Tree contain additional information that the non-connecting members do not require. For the example listed in Table 3.2, a corner piece, the member has three connection points. The connections of the corner piece can be seen in Figure 3.3. Each connection is assigned an associated number and annotated within the table. The connection is then assigned a member that it is connected to. For placement of the non-connecting member, a connection type is generated as "Pos" or "Neg". The "Pos" connection dictates that the member is attached by the first node of the body and a "Neg" type connects the member through the last body node.

The SimpleSat Model Tree contains a total of thirty-four members (see Appendix C), twelve of which are structural beams. Also included are eight corners, eight collars, two solar cross members, two solar supports, and two solar panels.

3.4.2.3 Model Construction

The construction of the complete FE model for SimpleSat is accomplished by utilizing the component FE models and the Model Tree. The nodal relationship within the component FE models is maintained, but the entire component model's nodal numbers are assigned global node numbers consecutively based on the placement of the members within the Model Tree with member number one's first body node having the value of one, continuing in that fashion until all member bodies are numbered. The connector nodes are numbered last to place all connectors at the end of the global matrices, as discussed in Chapter 2.

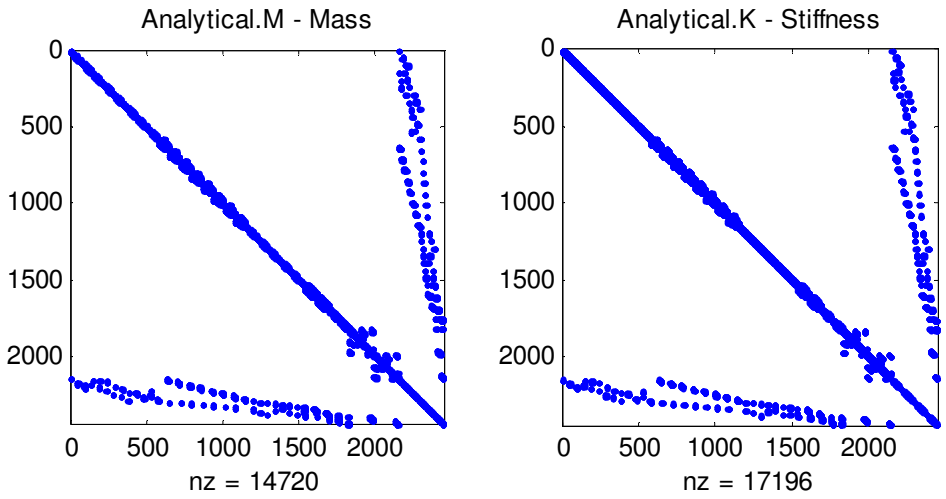
The process of adding the individual components to the compiled model utilizes a two path construction technique. The two segmented paths are the development of the graphical

and analytical models. The graphical model is not specifically required for analysis purposes and is only used within the Structural Dynamics Toolbox (SDT) to ensure the analytical model is correct. The analytical model has the global mass and stiffness matrices which are used for eigenvalue and eigenvector analysis.

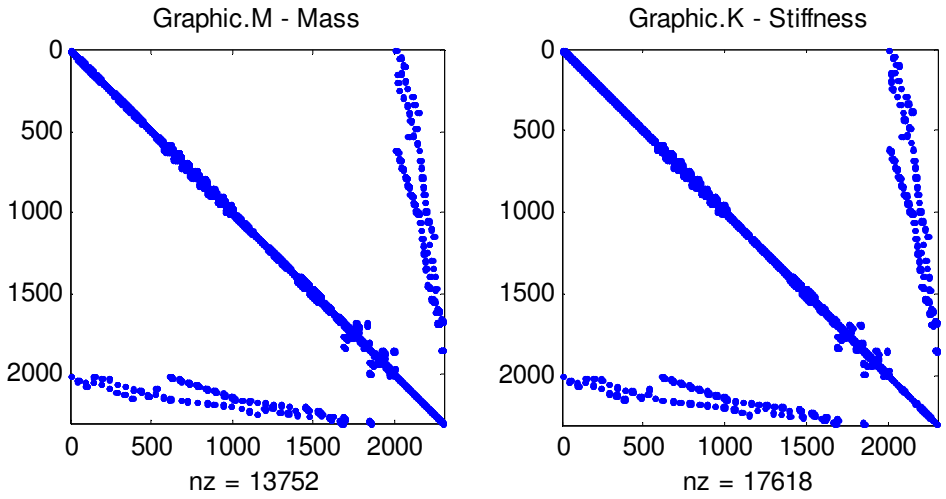
Although two significantly different paths are taken for analysis, all modifications to the nodal numbers and compilation of matrices is done coincidently to assure placement of each node within the models is consistent. Each member's stored model files are read in sequentially based on location within the Model Tree. The nodal characteristics are initially read in from the .Node file and the physical location is transformed based on the position vector. The nodal relationship is then modified to reflect the new nodal numbers. A matrix transformation is then required for the global mass, *M*, and stiffness, *K*, matrices for the specified node, again based on the position vector. These new values are then added to the system level Node, Elt, *M* and *K* files.

After all body submatrices are transformed, the connection nodes and relationships are then placed in the global matrices, all derived from the connection information in the Model Tree. During the renumbering process performed on the body submatrices, the locations of the key nodes are recorded. The connection nodes are placed within the Node, *M*, and *K* files. Based on the detailed connection information and location of the connection, the relationship between the connection node and the nodes it is connecting to is annotated within the Elt, *M* and *K* matrices.

The model assembly process is completed using a single MATLAB® script titled MatrixConst.m (Appendix C). The entire process is eased due to the symmetry of the *M* and *K* matrices and by the limited amount of information due to the sparse matrices. An additional section of code is used to fill the symmetric portion of the matrices because only the upper right triangle of the global M and K are processed within the matrix construction code. To ensure that the two processes produce the same result, a *M* and *K* matrix are produced from the graphical analysis and both are compared using the MATLAB® "Spy" command. Figure 3.8 displays the output of the "Spy" command for both the graphical and analytical results. From visual inspection of these matrices, one can see that the two methods match and nodal placement is maintained between the methods.



(a)



(b)

Figure 3.8: Mass and Stiffness Matrices for the (a) Analytical and (b) Graphical Methods

The nodal segregation and grouping of the member bodies is demonstrated quite well in Figure 3.8. The connection nodes and connecting matrices between the nodes and the member bodies is shown as well.

3.4.3 Results of SimpleSat Analysis

The successful modeling of the SimpleSat is determined by two criteria: timeline for computation and whether the model used is valid. For the timeline comparison, time checks will be made within the MATLAB® code. The validity of the model will be accomplished by comparing the SDT SimpleSat model with an additional computer generated model as well as experimental tests to determine the modal frequencies. Each has been outlined in the following sections.

3.4.3.1 Timeline Analysis

For comparison, four iterations of the model analysis were completed to determine time differences of the code segments. The first run utilizes the model with all components at the lowest resolution, resulting in a total of 384 nodes for the complete FE model. In the second iteration, the resolution of the solar panels was increased to the maximum resolution, from 27 to 45 nodes, increasing the total number of nodes to 420. The third run increases the resolution of two longitudinal structural beams to medium resolution and maintains the solar panels at high resolution. The total number of nodes within the third run is 600. The fourth and final iteration of the model was to increase all available resolutions to the medium resolution. This included the structural beams in all directions, the attachment collars, and the solar panels. The total node count increased by 900 resulting in a total number of 1500 nodes, or 9000 DOF.

Table 3.3 shows specific time information for each of the iterations. The development of the component FE models remains relatively the same for every configuration. The obvious, and expected, time constraint is the model assembly. By simply averaging the number of nodes used compared to the time of computation, we can use a figure for calculation of 1 to 1.5 seconds per node, due to the renumbering scheme required to make the model follow the established guidelines of nodal segregation approach. Table 3.3 does not depict the time required to establish the initial models and modify the overall model to include the changes

in resolution. That is because the construction of the models does not factor into the analysis time because they were created prior to Model Tree development, which is the foundation for this approach. The modification to the Model Tree is insignificant, as the time required to change a small number of characters and to resave the file is miniscule.

Table 3.3: Time Comparison Data for Four SimpleSat Iterations with Varying Complexity

SimpleSat Model Analysis		
Iteration #1 (384 Nodes)		
Action	Time Required (mm:ss.ss)	% of Total Time
Component FE Model Creation	00:12.63	5.54%
Model Assembly	03:30.08	92.20%
Eigenpair Analysis	00:05.15	2.26%
Total Time	03:47.86	
Iteration #2 (420 Nodes)		
Action	Time Required (mm:ss.ss)	% of Total Time
Component FE Model Creation	00:12.63	4.57%
Model Assembly	04:21.00	94.49%
Eigenpair Analysis	00:02.58	0.94%
Total Time	04:36.22	
Iteration #3 (600 Nodes)		
Action	Time Required (mm:ss.ss)	% of Total Time
Component FE Model Creation	00:12.87	2.46%
Model Assembly	08:27.09	97.00%
Eigenpair Analysis	00:02.83	0.54%
Total Time	08:42.79	
Iteration #4 (1500 Nodes)		
Action	Time Required (mm:ss.ss)	% of Total Time
Component FE Model Creation	00:12.58	0.40%
Model Assembly	52:09.06	99.42%
Eigenpair Analysis	00:05.80	0.18%
Total Time	52:27.44	

It is noted that the size of the models used in the SimpleSat demonstration are small compared to a complex FE model, which can easily consist of multi-millions of DOF. The size of the assembled model directly effects the time required for analysis computation. As the number of nodes increases, the dimension of *M* and *K* increase by a factor of six, and therefore effect the time required to compute the eigenpairs increases on the order of the number of nodes cubed. As the size of the models increase, the eigen analysis will quickly become the dominant time consumer.

All of the iterations of the SimpleSat model were completed using a Dell 700M notebook computer with an Intel 1594 MHz Centrino™ processor with 1024 MB of RAM. This machine was running Microsoft® Windows® XP Home Edition. Since most computations are floating point, the analysis process would be significantly faster given a computer with a faster floating point unit.

3.4.3.2 Validation of SimpleSat Model

Validation of a model can entail many different tests to prove that the model matches the characteristics of the actual hardware. For our comparison, only the modal frequencies will be compared. This limits the true validity of this model, but is adequate to determine whether the results achieved are acceptable. Three sets of data are compared for the SimpleSat validity determination, the SDT modal frequencies, modal frequencies determined by a Nastran model, and frequency determination by experimentation on the machined hardware as seen in Table 3.4. The background data for which Table 3.4 is developed is included in Appendix B.

The experimental results are are derived from a series vibration response tests on the SimpleSat hardware. We placed the assembled SimpleSat model in free space, hanging from a simple damperless rubberband setup, and struck the model at various locations. The response was recoded via two accelerometers and then plotted with MATLAB. The results in Table 3.4 depict the average of the similar modal frequencies. The complete results from our testing can be found in Appendix B.3.

Table 3.4: Comparison of Modal Frequency Determinations

Modal Frequency Comparison		
SDT	Nastran	Experimental
Frequency (Hz)	Frequency (Hz)	Frequency (Hz)
158.7	71.28	72.89
163.2	82.16	86.89
174.1	133.12	92.81
192.5	209.38	137.12
199.9	212.54	198.58

Table 3.4 is a simple comparison of the first five modal frequencies from the three separate modeling methods. Basic interpretation of the data indicates that the three methods utilized for model construction provide vastly differing results. The experimental and Nastran results demonstrate concurrence within four of the five modes, differing with the exclusion of the third mode within Nastran. The deleted node is most likely due to the modeling process within Nastran. The Nastran model developed only utilized the Solar Panel and the Solar Support, ignoring the effect the remainder of the system provided.

The SDT results do not agree in the low frequencies, but are closer near mode 5, which is merely coincidental due to the mode shape differences. Appendix B.1 shows the mode shapes from the SDT analysis. Inspection of these shows that there is a major flaw within the SDT processes. The mode shapes demonstrate a non-symmetric vibration throughout all frequencies, providing incorrect results due to no off axis boundary conditions. The errors within SDT may also be associated with the toolbox's unit conversion factors. The model is developed in US units, specifically inches and pounds, and SDT converts the units to SI. From basic code manipulation, it has been determined that the conversion is flawed, and must be corrected before adequate results can be achieved.

4 Conclusions and Future Work

The purpose of the research presented was to prove that with a finite element (FE) model development paradigm shift, a finalized satellite FE model could be produced within the Operational Responsive Space (ORS) assembly timeframe. Within the major research goal were several smaller objectives, including the development of a series of models, both hardware and FE, to test the concept as well as the development of MATLAB® code to assemble the global FE model.

4.1 Conclusions

The results from the analysis presented shows that our research was partially successful. We took the first of many steps required to generate a method to place FE modeling within the ORS assembly window. We proved that, if a FE model is constructed following the dedicated interface rules, a global FE model can be assembled within the ORS program timeline. The nodal segregation approach demonstrated that by utilizing a constant distance between a connector node and the associated body node, the interface of the modeled component with the assembled FE model did not change, providing the opportunity to exchange refined models with very little impact. The result is an assembled model that can be refined on-the-fly.

We demonstrated that by producing a variety of models for each component prior to any assembly activity, the majority of the work required for global FE model assembly is accomplished before any need is defined, significantly reducing the amount of time and effort required to assemble the global model. The work remaining in assembly is limited to the connectivity definition of each component used.

The MATLAB® routine developed to build the component models, assemble the global model, and perform the eigenpair analysis showed that the nodal segregation approach to FE development will significantly improve the model development time. With the completion of the code and the success of the trials performed, the developed routine is considered to be a “proof-of-concept’ of the nodal segretation approach.

The potential within the application of our nodal segregation approach is limitless within the modeling and simulation of component based systems, to include the astronautical, aeronautical, and automotive industries. The research performed is vital to the success of the ORS program, and will be beneficial to many other Air Force specific programs.

4.2 Future Work

While much work has been completed in the development of the FE model production methodology as presented in this paper, a great deal of work remains incomplete. The assembly process has proved successful, but the timeline and effort required can be reduced further. The modeling within Structural Dynamics Toolbox (SDT) was successful to a limited extent, but, as mentioned previously, must be refined. As effective as SimpleSat is in demonstrating the basic concepts of Plug-and-Play componentry, it would be ideal to begin modeling the actual hardware components currently being developed by AFRL/VS.

4.2.1 Approach Refinement

The nodal segregation approach was widely successful in the proof of concept. However, during the analysis trials for SimpleSat, the majority of the processing time was spent assembling the global FE model. We can further optimize the assembly process by moving additional model development prior to assembly. Because the majority of the component configurations are known, the development of additional component FE models representing all known model rotations could be accomplished. The existence of the rotated models can eliminate the need to rotate the placement vectors and matrices for each node, removing a significant portion of the computing time.

4.2.2 Model Complexity

The models created within SDT are limited with regard to practical modeling of the SimpleSat componentry. To ensure compatibility of each component's FE model with the method processes, each component was modeled with essentially a one-dimensional

mentality. In every instance, the models developed are a single string of nodes, no more than one node deep. A system-wide overhaul of the models to increase nodal density, mesh refinement, and to include plate and solid elements would significantly increase the realism of the assembled model. The complete validation of the models with respect to the SimpleSat hardware will also significantly improve results.

Due to the results found from the comparison of the model analysis to the actual experimental values, the SimpleSat SDT model holds no validity to the actual system. The moments of inertial (MOI) need to be verified as correct within the models. The stiffness characteristics within the model should also be checked.

4.2.3 Analysis Approaches

To reduce the dimension of the component mass and stiffness matrices, a component mode synthesis (CMS) approach would be used. Using a CMS or Craig-Bampton [4] approach, we could project the component mass and stiffness matrices onto their respective truncated eigenspaces. A reduction in dimension of the system matrices of a factor of several thousand is fairly common for large FE models. This approach could be used in the ORS approach.

4.2.4 Modeling Flight Hardware

The next logical step in applying nodal segregation to the Plug-and-Play structural environment is to model the hardware being developed for the Operational Responsive Space (ORS) program. Currently, the Air Force Research Laboratory's Space Vehicle Division (AFRL/VS) is working with a multitude of contractors to begin development of basic flight hardware for this effort. To model these components and assemble a mock system would be the final step to proving the methodology presented is legitimate.

APPENDICES

Appendix A: SimpleSat Manufacturing Drawings

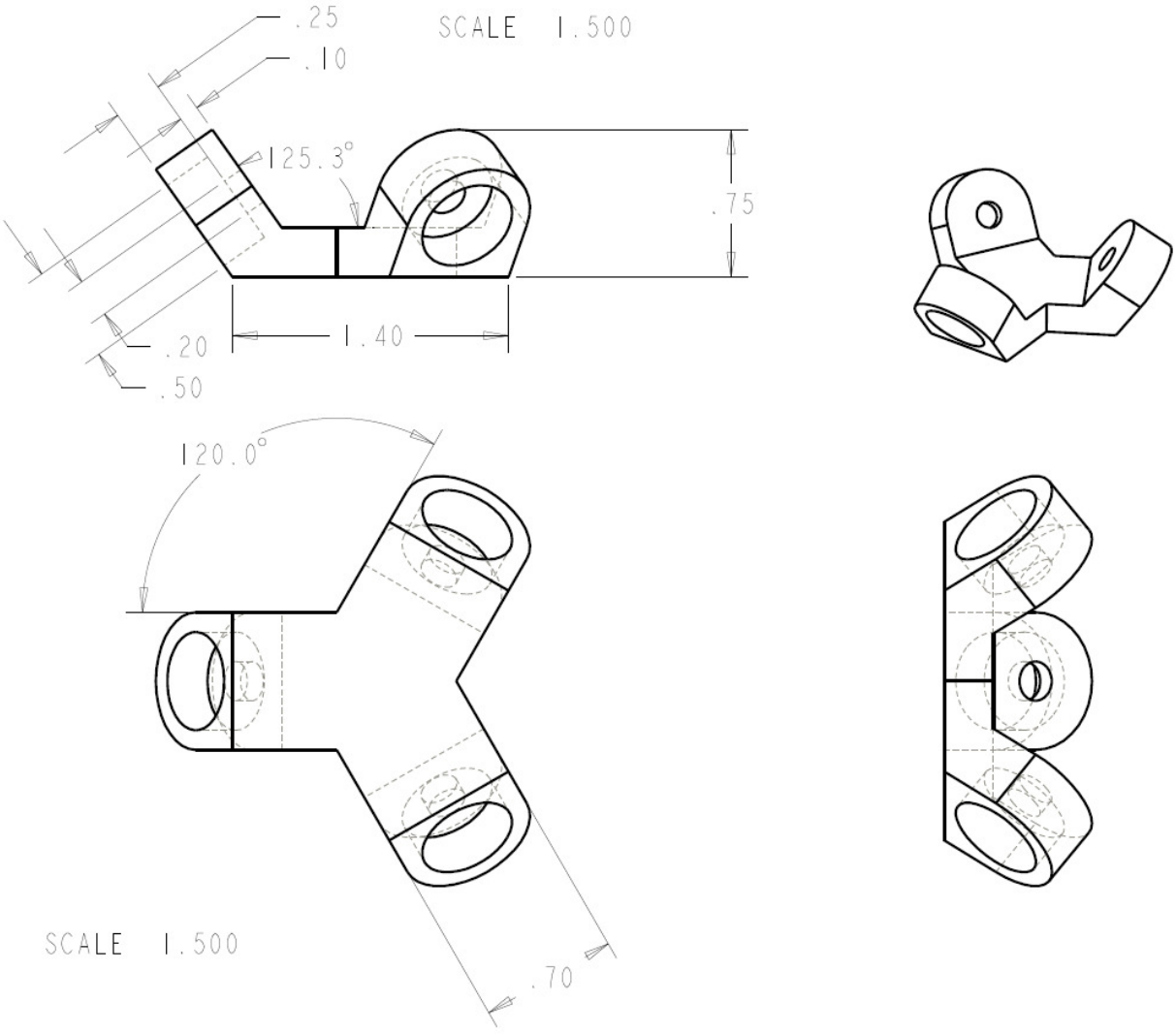


Figure A.1: Attachment Point Drawing

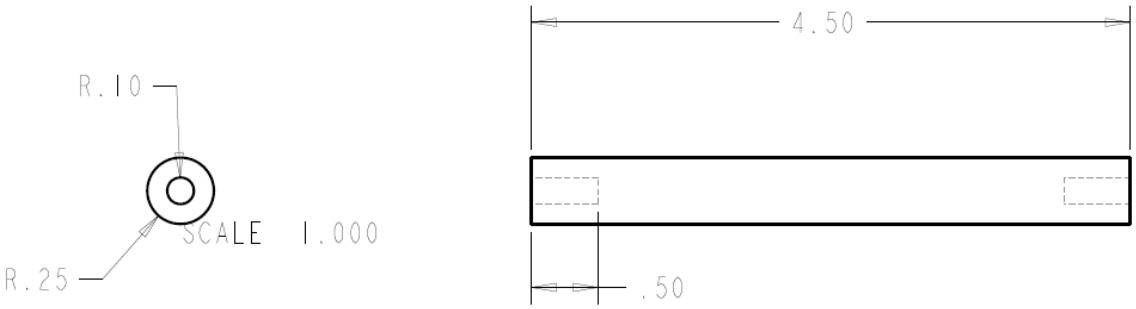


Figure A.2: Four Inch Beam Drawing

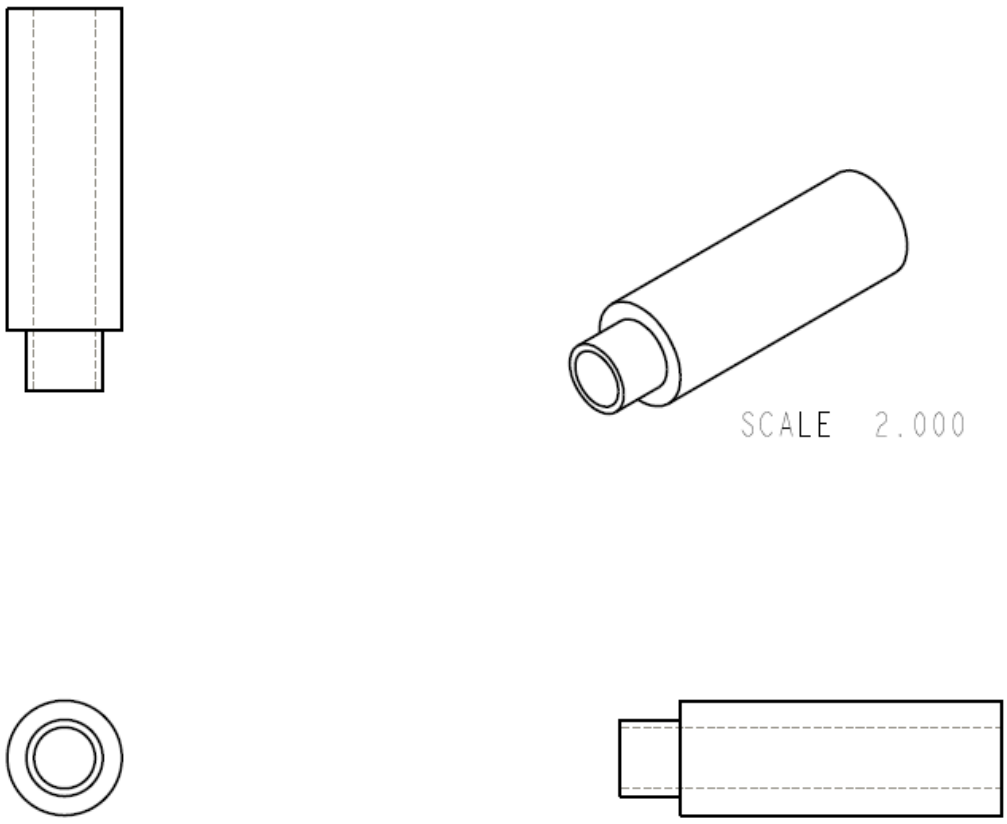


Figure A.3: Solar Attachment Collar Drawing

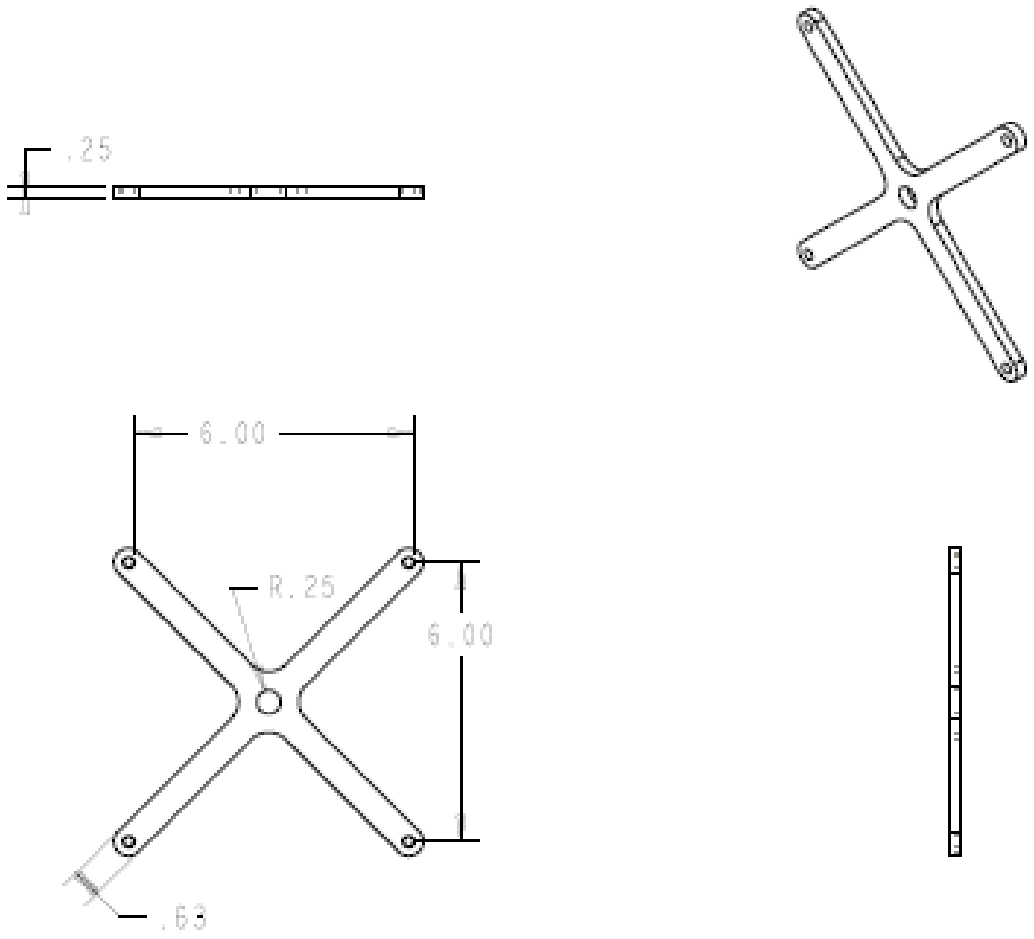


Figure A.4: Solar Assembly Attachment Cross Member Drawing

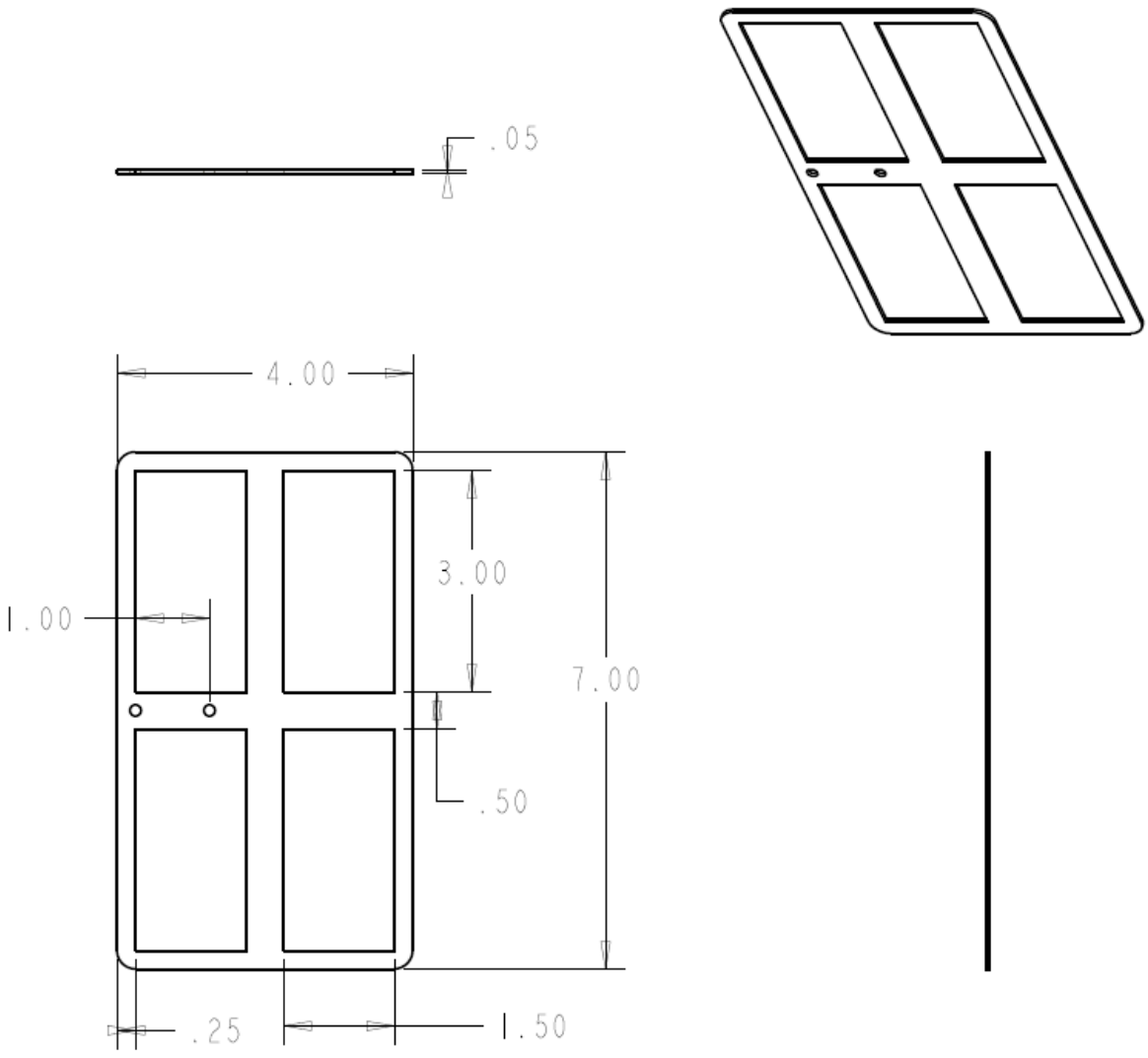


Figure A.5: Solar Panel Drawing

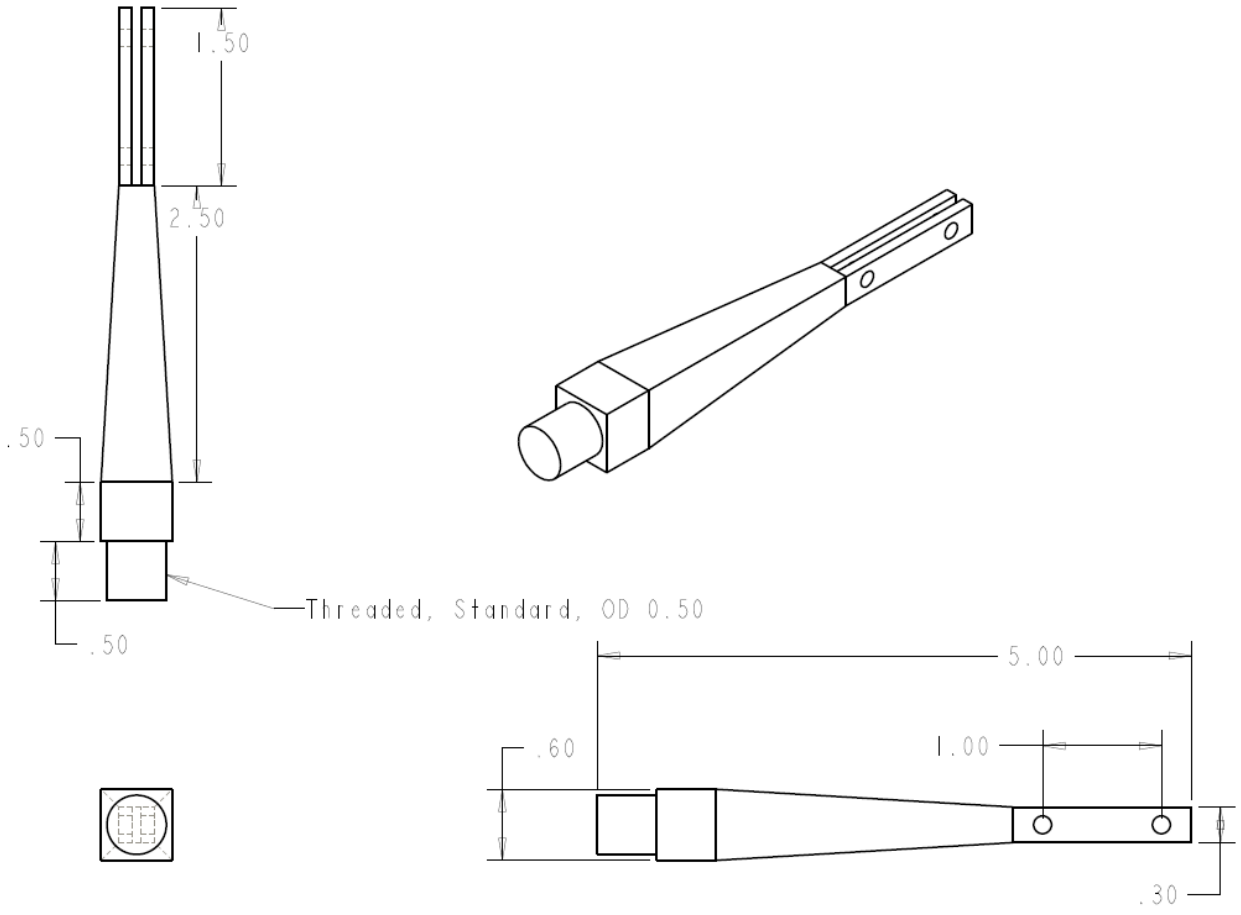
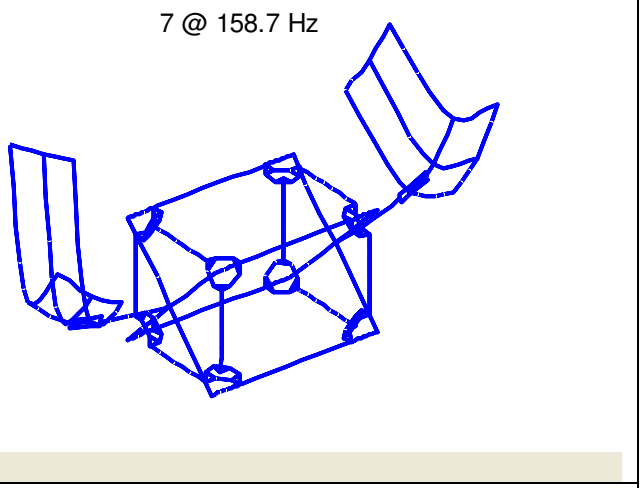
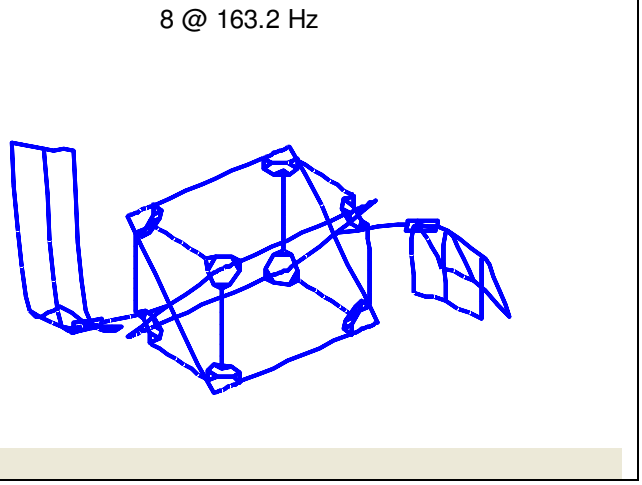
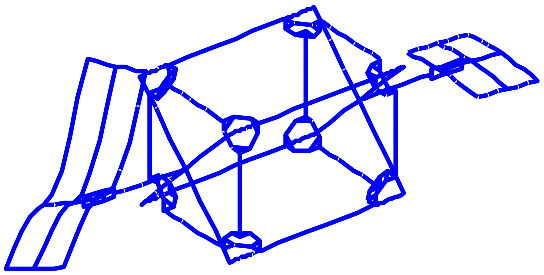
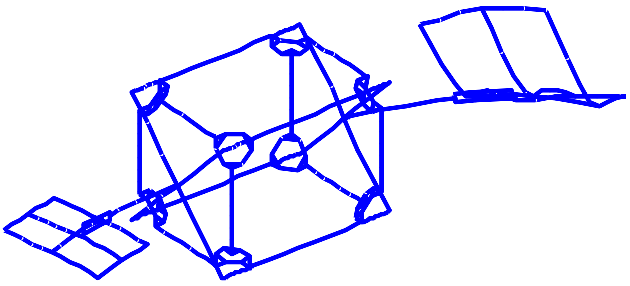
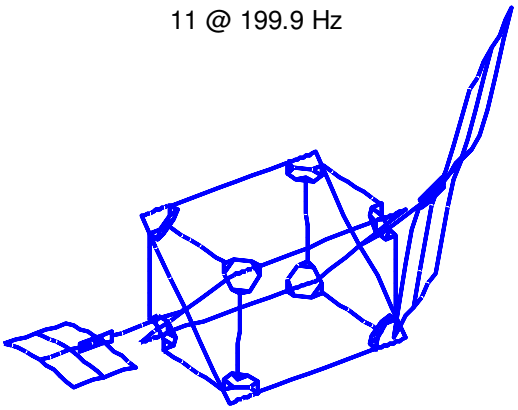


Figure A.6: Solar Panel Support Drawing

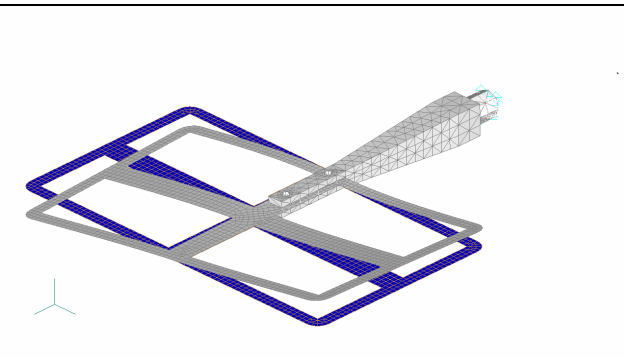
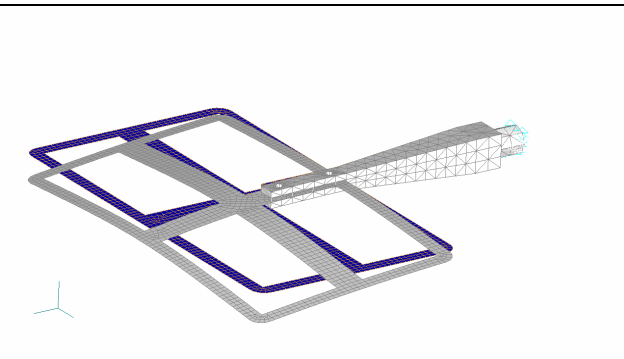
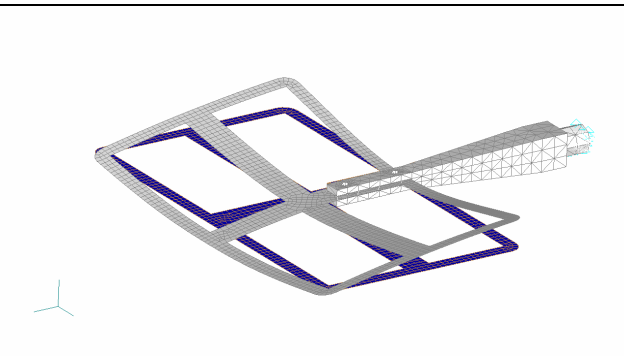
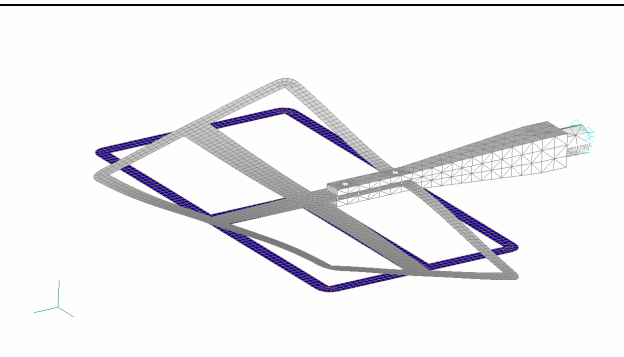
Appendix B: Validation of the SimpleSat FE Model

Appendix B.1: SDT Modal Frequency Analysis

Mode 1 – 158.7	
Mode 2 – 163.2	

Mode 3 – 174.1	9 @ 174.1 Hz  current object info
Mode 4 – 192.5	10 @ 192.5 Hz  current object info
Mode 5 – 199.9	11 @ 199.9 Hz  current object info

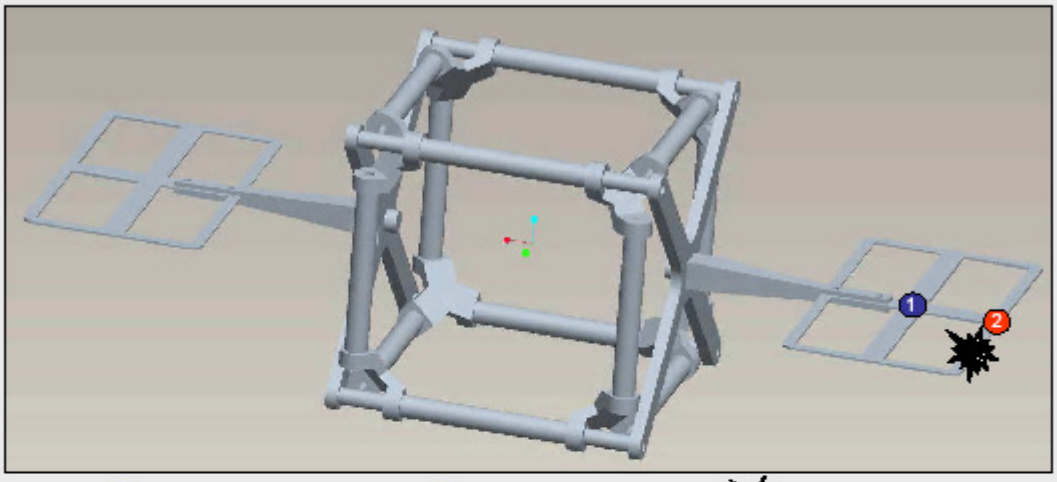
Appendix B.2: Nastran Modal Frequency Analysis

Mode 1 – 71.28 Hz	
Mode 2 – 82.16 Hz	
Mode 3 – 133.12 Hz	
Mode 4 – 209.38 Hz	

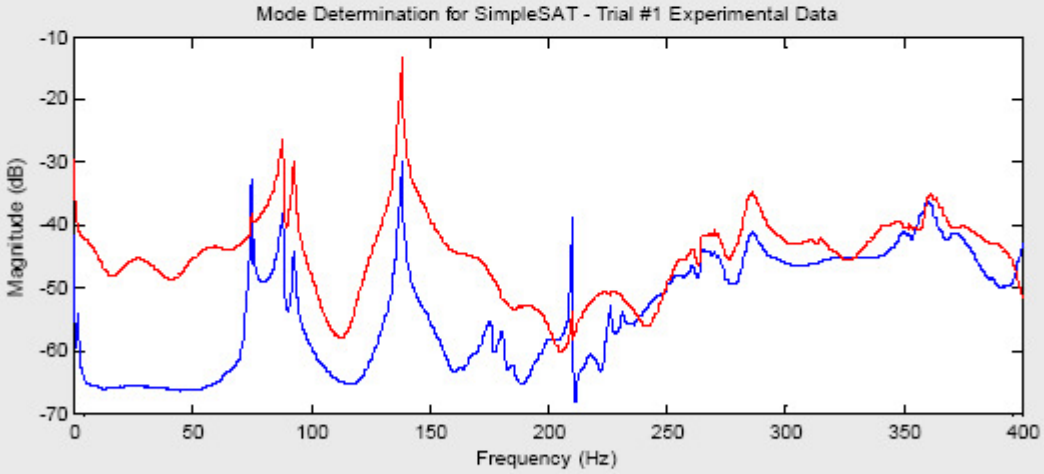
Appendix B.3: Experimental Modal Frequency Analysis

SimpleSat Vibrational Analysis

Trial #1



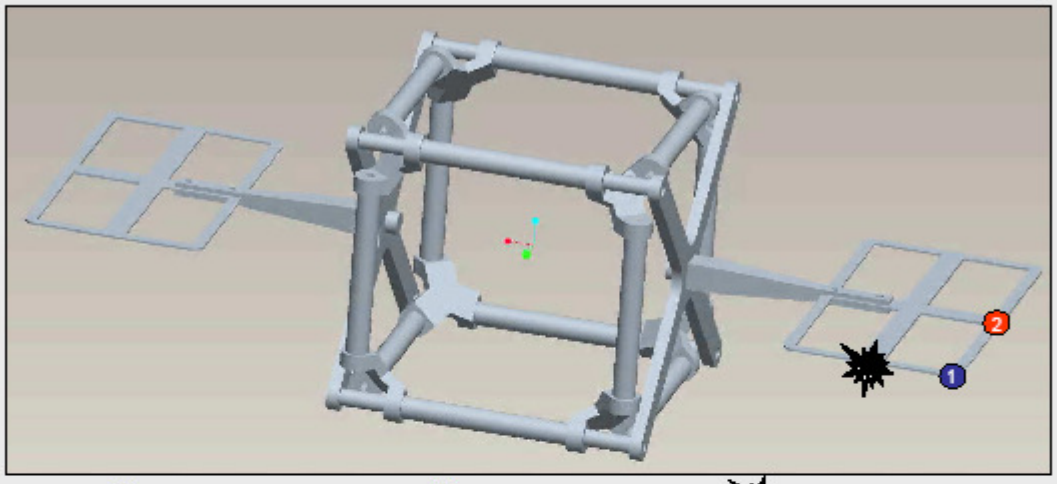
1 Accelerometer #1 Location 2 Accelerometer #2 Location Strike Location



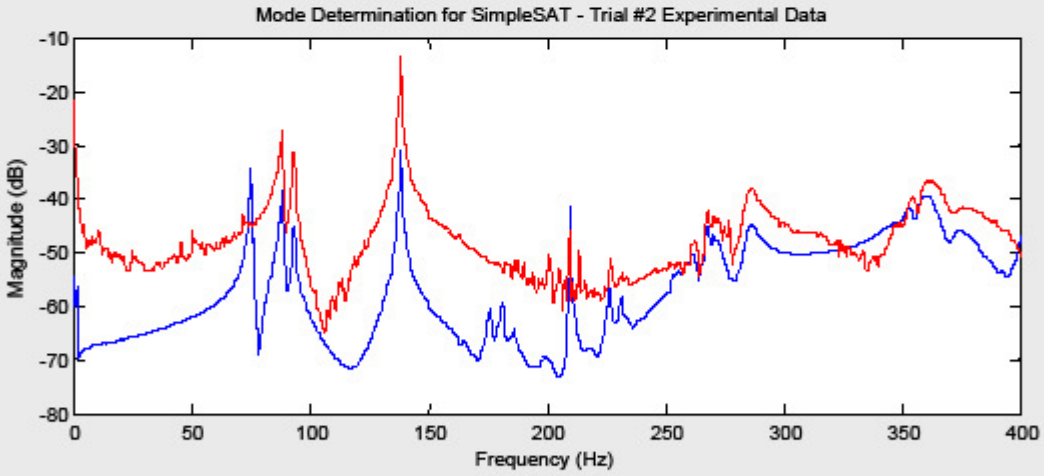
Mode Frequency Analysis										
Accelerometer	1st Mode		2nd Mode		3rd Mode		4th Mode		5th Mode	
	dB	Hz	dB	Hz	dB	Hz	dB	Hz	dB	Hz
1	-32.77	75	-38.19	88	-44.14	93	-29.99	138.5	-38.75	210
2	-38.16	75	-26.41	88	-29.73	93	-13.3	138.5	-53.68	210

SimpleSat Vibrational Analysis

Trial #2



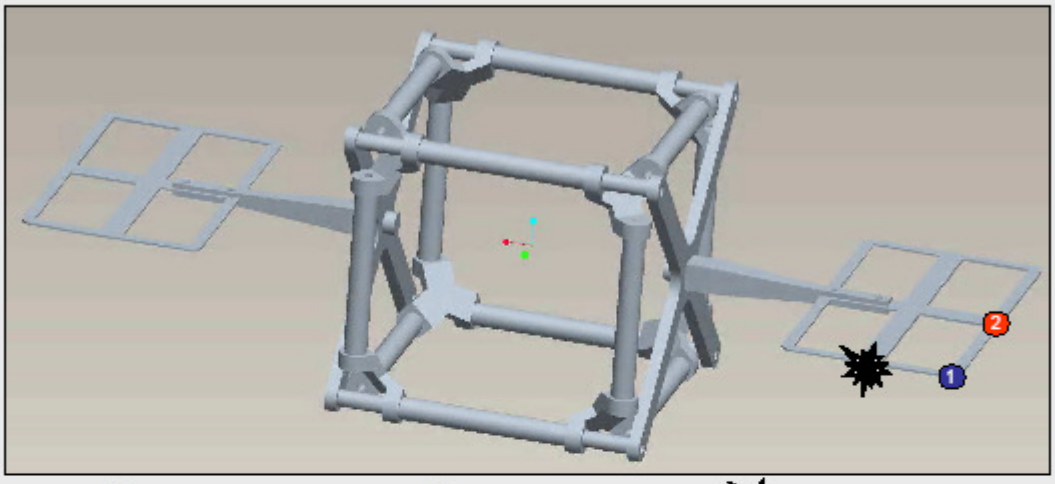
1 Accelerometer #1 Location 2 Accelerometer #2 Location Strike Location



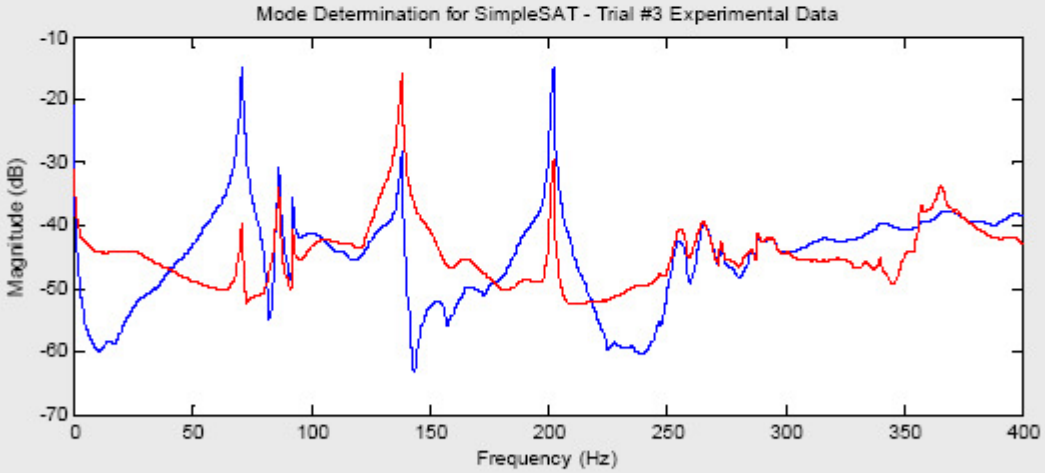
Mode Frequency Analysis										
Accelerometer	1st Mode		2nd Mode		3rd Mode		4th Mode		5th Mode	
	dB	Hz	dB	Hz	dB	Hz	dB	Hz	dB	Hz
1	-34.45	75	-38.61	88.5	-45.23	93	-30.83	138.5	-41.51	210
2	-4.91	71.5	-27.53	88.5	-31.10	93	-13.76	138.5	-46.00	210

SimpleSat Vibrational Analysis

Trial #3



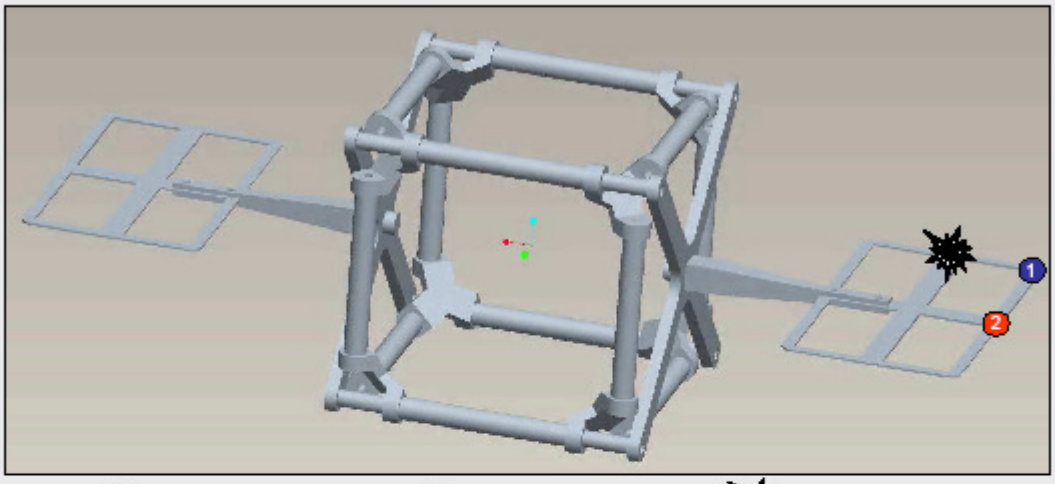
1 Accelerometer #1 Location 2 Accelerometer #2 Location Strike Location



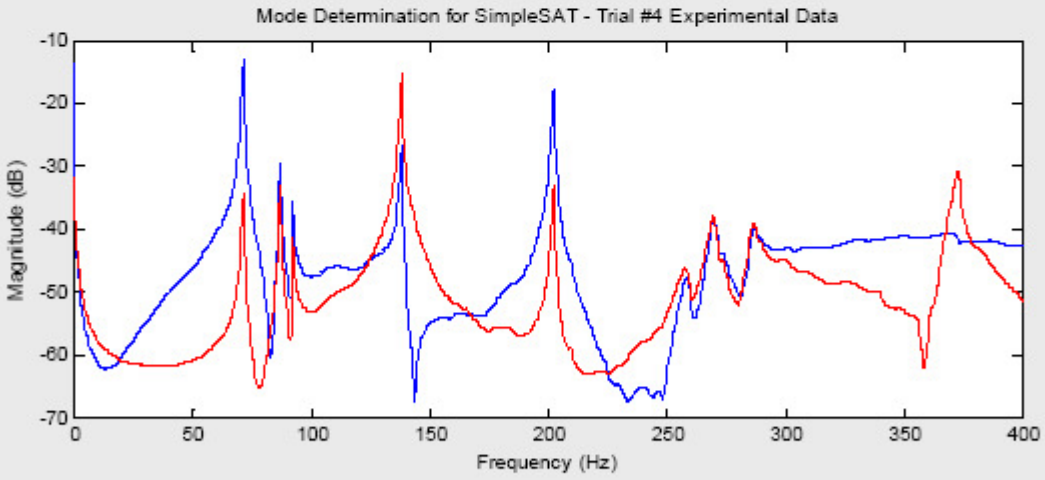
Mode Frequency Analysis										
Accelerometer	1st Mode		2nd Mode		3rd Mode		4th Mode		5th Mode	
	dB	Hz	dB	Hz	dB	Hz	dB	Hz	dB	Hz
1	-14.95	71	-30.71	86.5	-35.38	92.5	-28.12	138	-14.86	202
2	-39.66	71	-33.99	86.5	-40.43	92.5	-16.02	138	-29.42	202

SimpleSat Vibrational Analysis

Trial #4



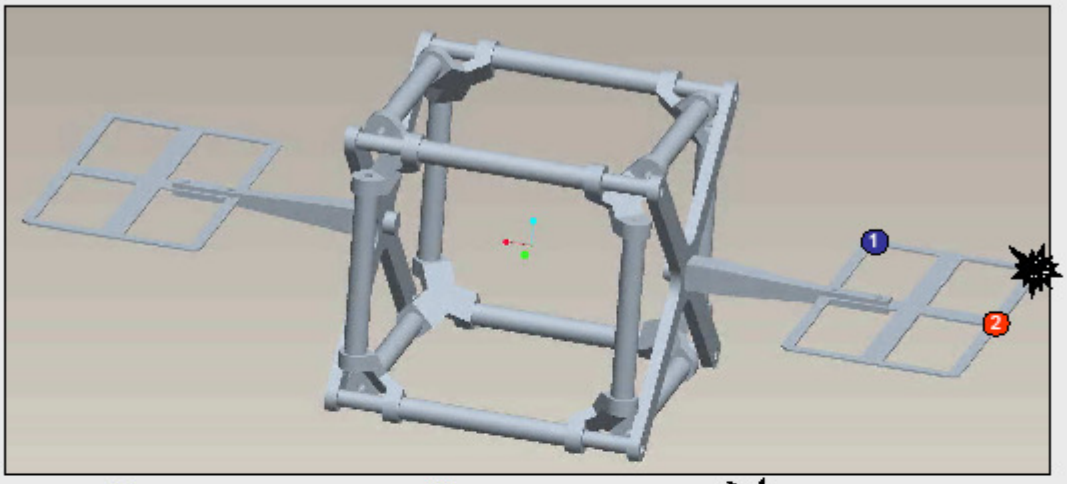
1 Accelerometer #1 Location 2 Accelerometer #2 Location Strike Location



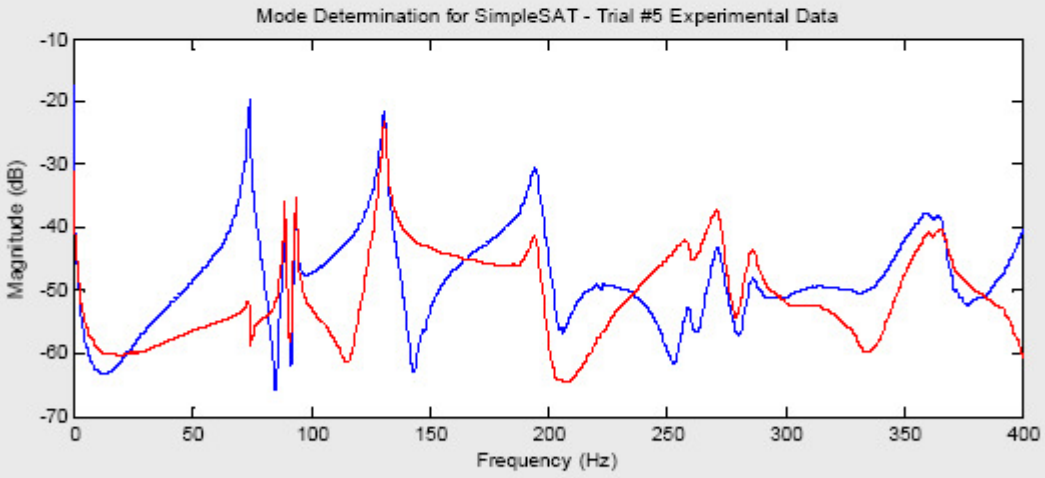
Mode Frequency Analysis										
Accelerometer	1st Mode		2nd Mode		3rd Mode		4th Mode		5th Mode	
	dB	Hz	dB	Hz	dB	Hz	dB	Hz	dB	Hz
1	-12.91	71.5	-29.42	87	-35.84	92.5	-26.62	138	-17.55	202
2	-34.14	71.5	-33.25	87	-41.36	92.5	-15.43	138	-33.26	202

SimpleSat Vibrational Analysis

Trial #5



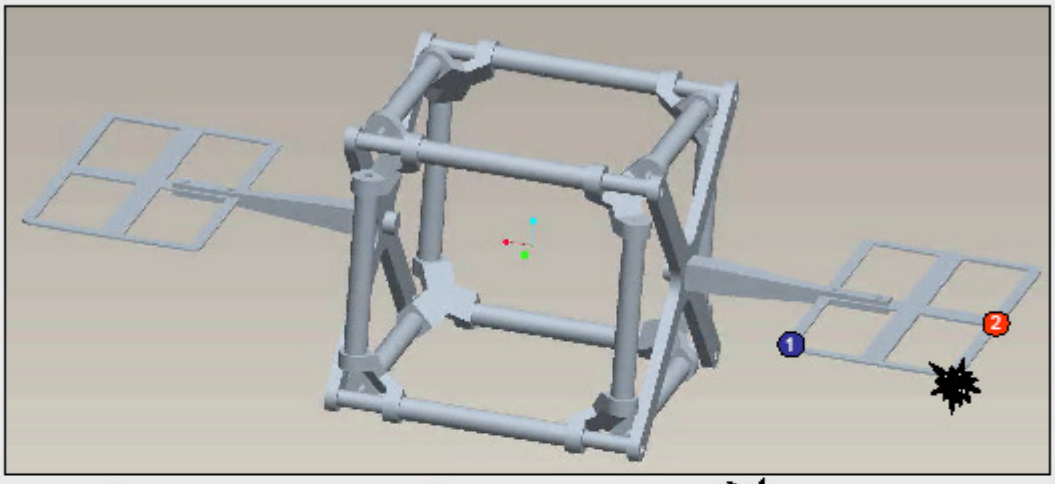
1 Accelerometer #1 Location 2 Accelerometer #2 Location Strike Location



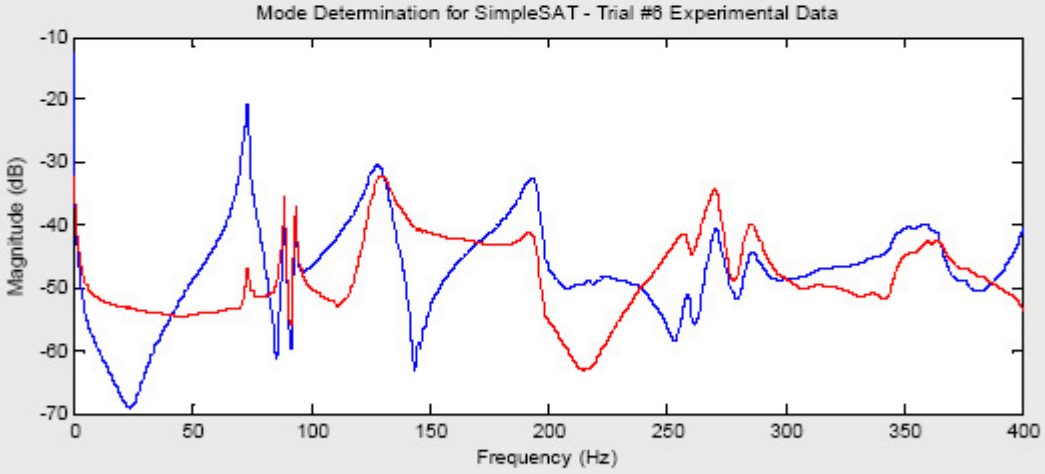
Mode Frequency Analysis										
Accelerometer	1st Mode		2nd Mode		3rd Mode		4th Mode		5th Mode	
	dB	Hz	dB	Hz	dB	Hz	dB	Hz	dB	Hz
1	-19.69	74	-37.11	89	-36.37	93.5	-21.49	131	-30.46	194.5
2	-51.87	73.5	-35.65	89	-35.12	93.5	-32.12	131	-41.32	194

SimpleSat Vibrational Analysis

Trial #6



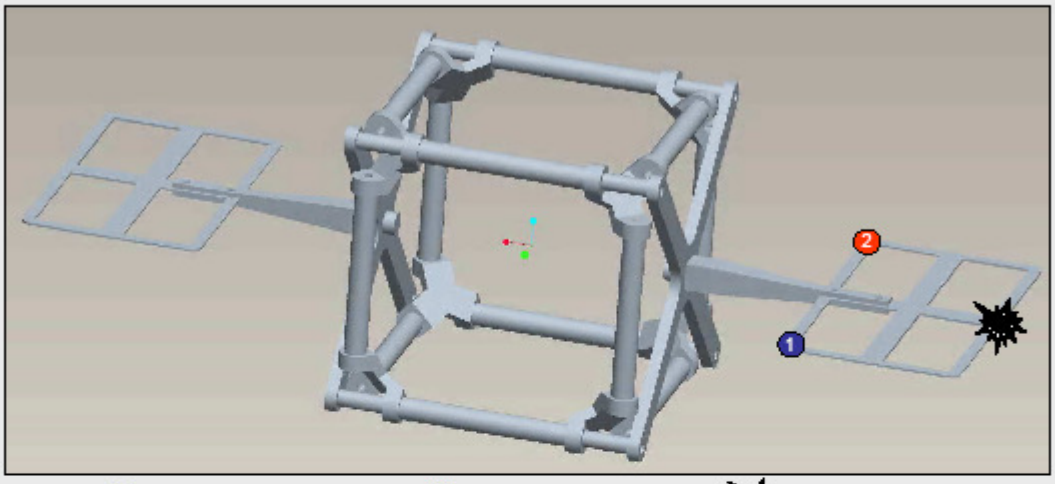
1 Accelerometer #1 Location 2 Accelerometer #2 Location Strike Location



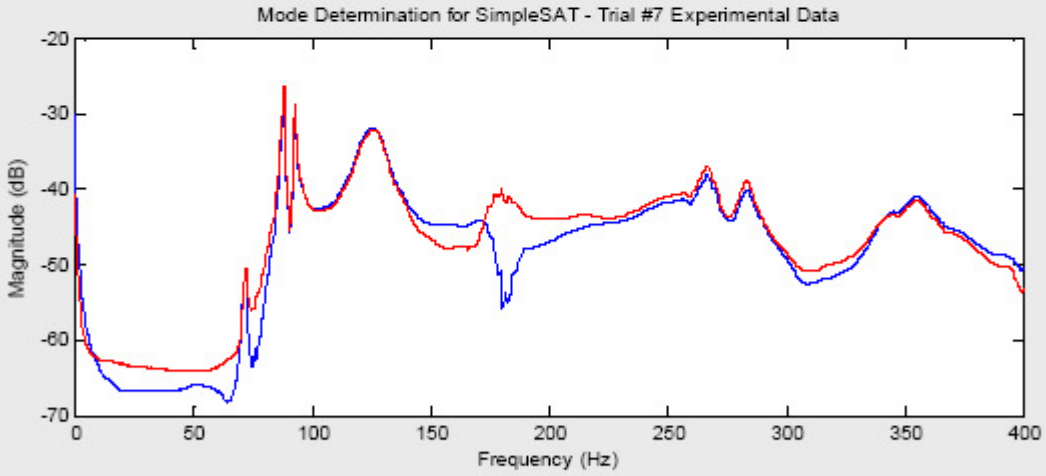
Mode Frequency Analysis										
Accelerometer	1st Mode		2nd Mode		3rd Mode		4th Mode		5th Mode	
	dB	Hz	dB	Hz	dB	Hz	dB	Hz	dB	Hz
1	-20.56	73	-37.71	88.5	-38.04	93.5	-30.44	128	-32.51	193
2	-46.71	73	-35.43	88.5	-37.05	93.5	-32.15	129.5	-41.27	192

SimpleSat Vibrational Analysis

Trial #7



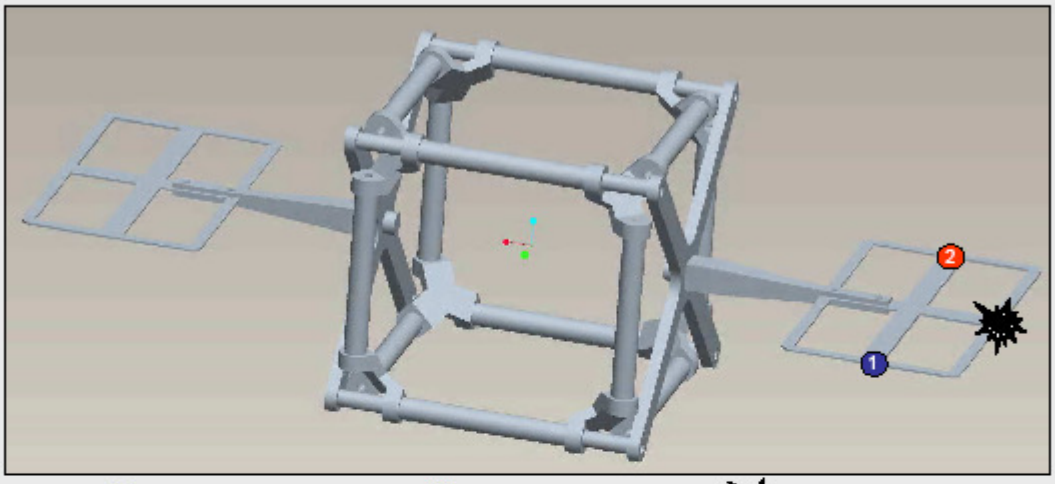
1 Accelerometer #1 Location 2 Accelerometer #2 Location Strike Location



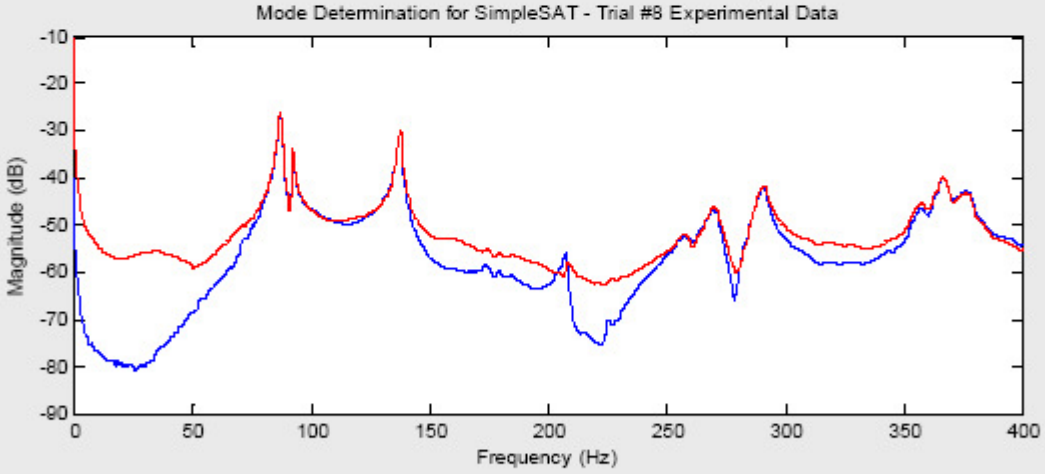
Mode Frequency Analysis										
Accelerometer	1st Mode		2nd Mode		3rd Mode		4th Mode		5th Mode	
	dB	Hz	dB	Hz	dB	Hz	dB	Hz	dB	Hz
1	-51.45	71.5	-27.59	88	-29.46	93	-31.87	125.5	-44.07	171
2	-50.37	72	-26.31	88	-28.59	93	-32.14	125.5	-39.80	180

SimpleSat Vibrational Analysis

Trial #8



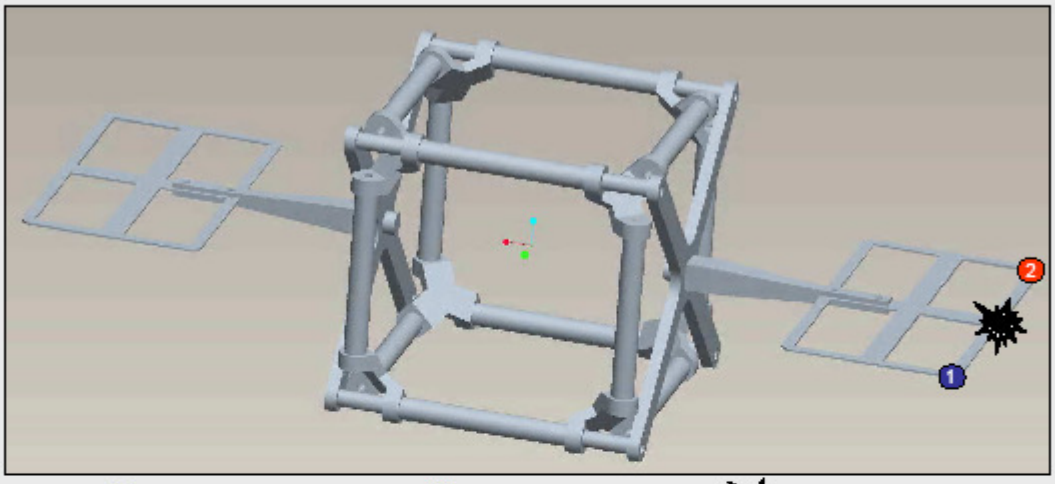
1 Accelerometer #1 Location 2 Accelerometer #2 Location Strike Location



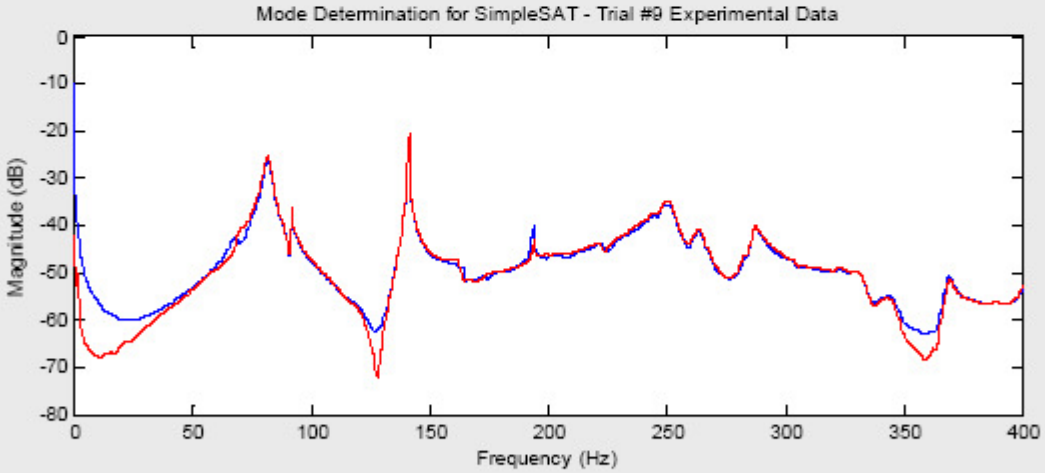
Mode Frequency Analysis										
Accelerometer	1st Mode		2nd Mode		3rd Mode		4th Mode		5th Mode	
	dB	Hz	dB	Hz	dB	Hz	dB	Hz	dB	Hz
1			-26.53	87	-33.63	92.5	-30.50	137.5	-55.83	207.5
2			-26.24	87	-33.45	92.5	-30.08	137.5	-57.89	208

SimpleSat Vibrational Analysis

Trial #9



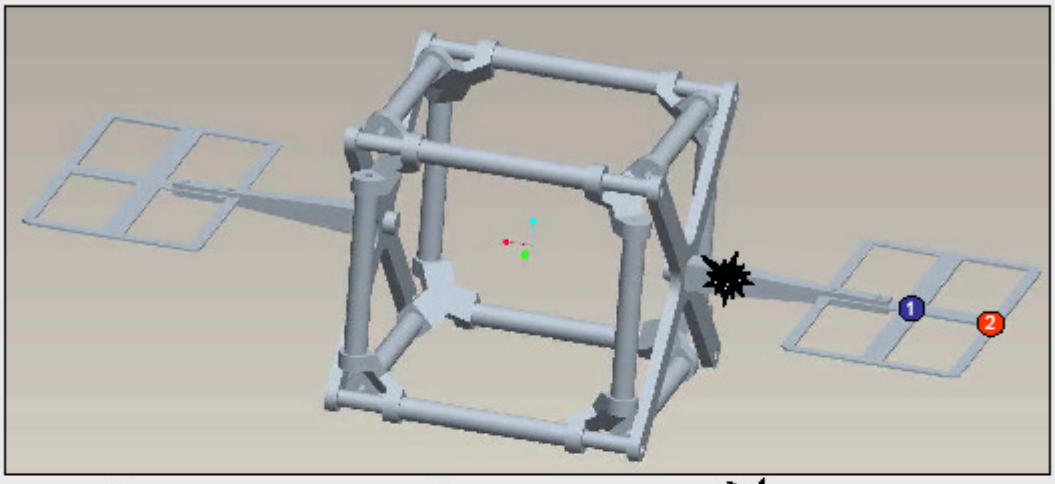
1 Accelerometer #1 Location 2 Accelerometer #2 Location Strike Location



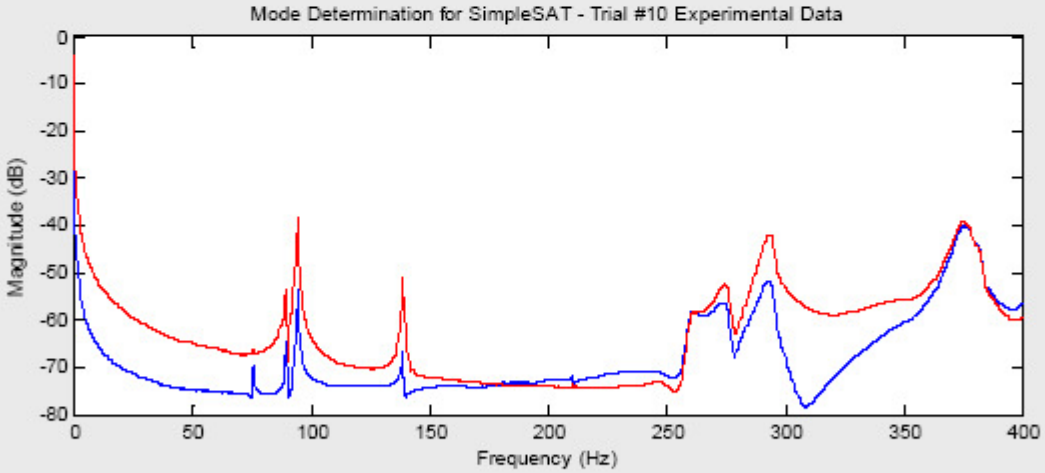
Mode Frequency Analysis										
Accelerometer	1st Mode		2nd Mode		3rd Mode		4th Mode		5th Mode	
	dB	Hz	dB	Hz	dB	Hz	dB	Hz	dB	Hz
1			-26.00	82	-36.72	92	-20.92	141.5	-40.03	194
2			-25.24	82	-36.00	92	-20.72	141.5	-42.97	194

SimpleSat Vibrational Analysis

Trial #10



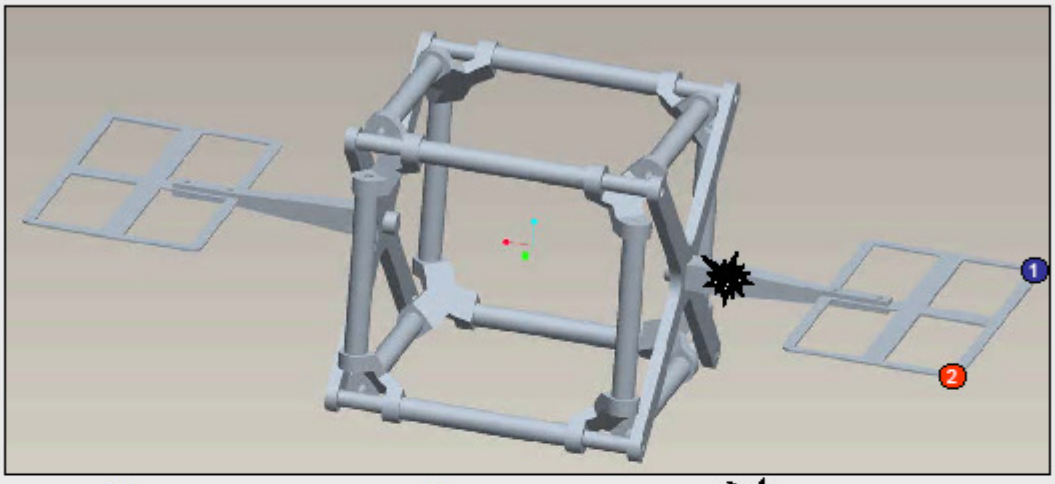
1 Accelerometer #1 Location 2 Accelerometer #2 Location Strike Location



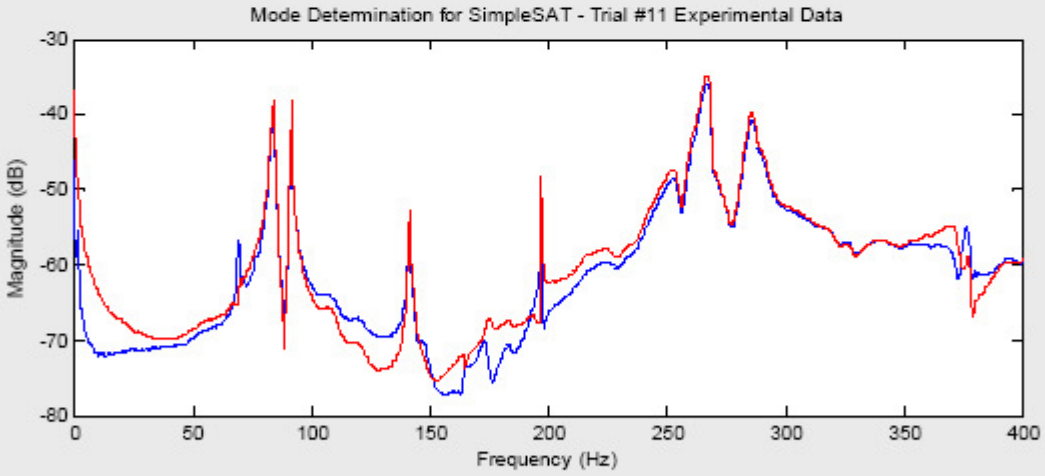
Mode Frequency Analysis										
Accelerometer	1st Mode		2nd Mode		3rd Mode		4th Mode		5th Mode	
	dB	Hz	dB	Hz	dB	Hz	dB	Hz	dB	Hz
1	-69.64	75.5	-84.89	89.5	-53.79	94.5	-66.70	138.5	-71.87	210
2	-66.41	75.5	-53.51	89.5	-38.48	94.5	-50.94	138.5	-73.34	210

SimpleSat Vibrational Analysis

Trial #11



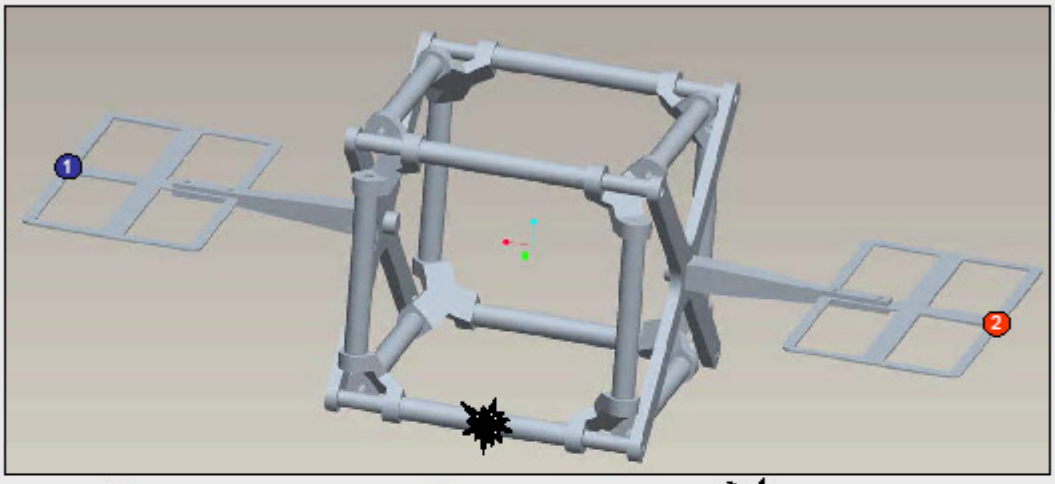
1 Accelerometer #1 Location 2 Accelerometer #2 Location Strike Location



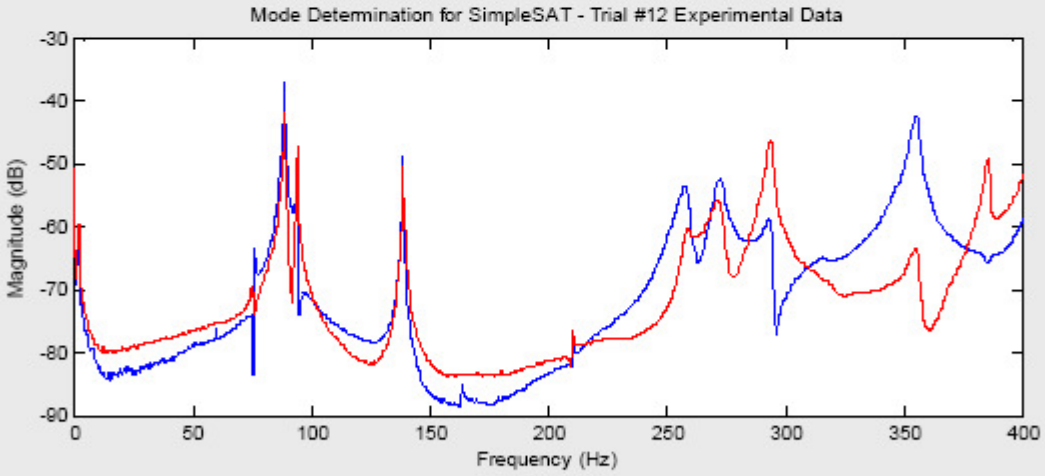
Mode Frequency Analysis										
Accelerometer	1st Mode		2nd Mode		3rd Mode		4th Mode		5th Mode	
	dB	Hz	dB	Hz	dB	Hz	dB	Hz	dB	Hz
1	-56.91	69.5	-39.07	84	-38.78	92	-53.09	141.5	-48.94	197
2	-61.96	70	-38.13	84	-37.96	92	-52.92	141.5	-48.22	197

SimpleSat Vibrational Analysis

Trial #12



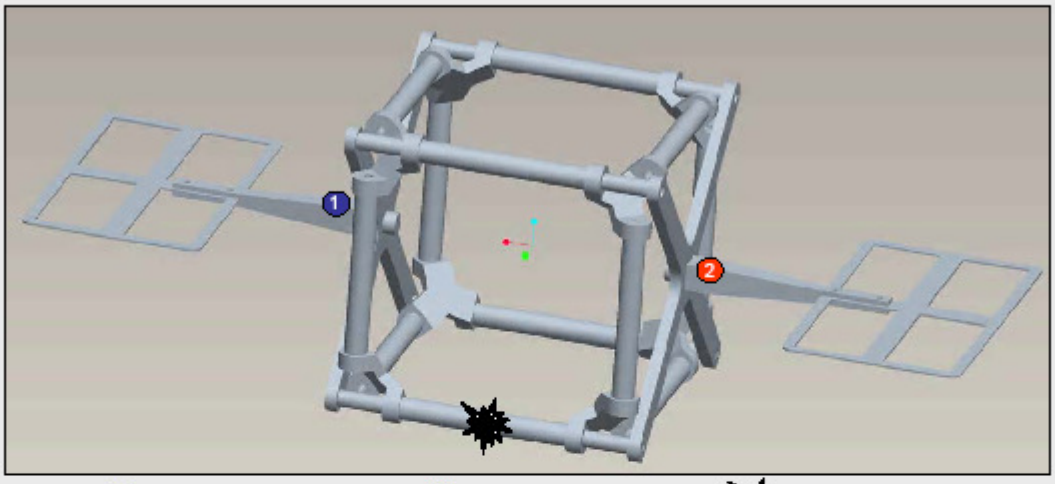
1 Accelerometer #1 Location 2 Accelerometer #2 Location Strike Location



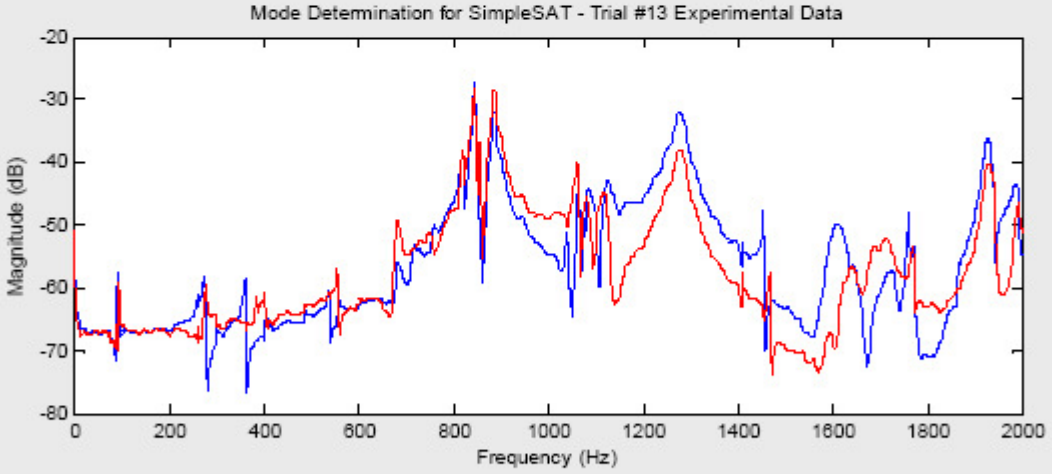
Mode Frequency Analysis										
Accelerometer	1st Mode		2nd Mode		3rd Mode		4th Mode		5th Mode	
	dB	Hz	dB	Hz	dB	Hz	dB	Hz	dB	Hz
1	-63.38	76	-36.83	88.5	-51.57	94	-48.67	138.5	-77.76	210.5
2	-70.55	76	-41.65	88.5	-47.16	94	-50.49	138.5	-76.38	210

SimpleSat Vibrational Analysis

Trial #13



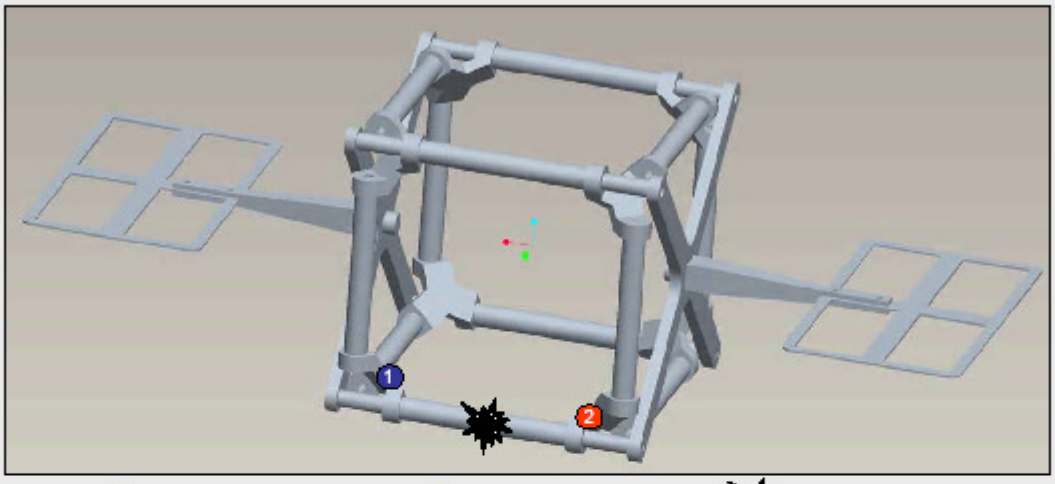
1 Accelerometer #1 Location 2 Accelerometer #2 Location Strike Location



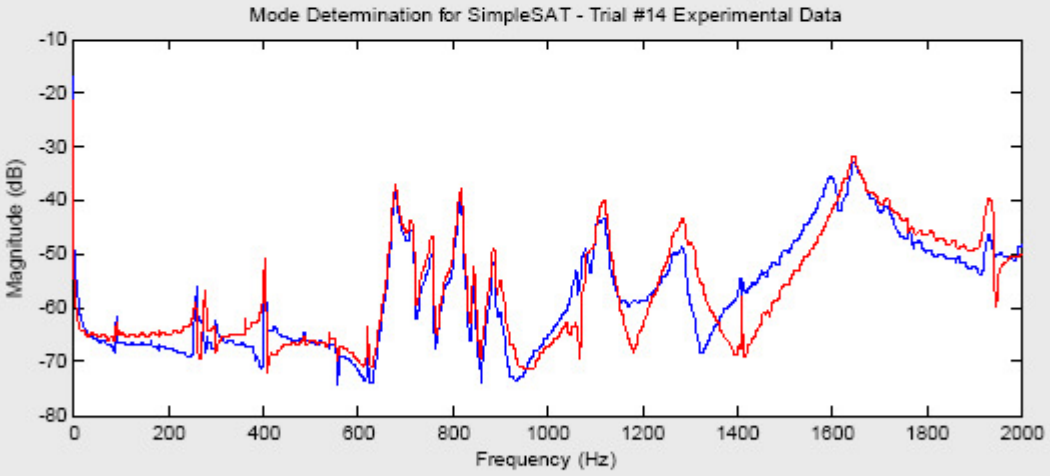
Mode Frequency Analysis										
Accelerometer	1st Mode		2nd Mode		3rd Mode		4th Mode		5th Mode	
	dB	Hz	dB	Hz	dB	Hz	dB	Hz	dB	Hz
1	-57.49	90	-58.33	275	-58.62	360	-27.3	845	-31.9	885
2	-59.01	95	-59.63	277.5	-60.64	402.5	-28.15	845	-28.4	885

SimpleSat Vibrational Analysis

Trial #14



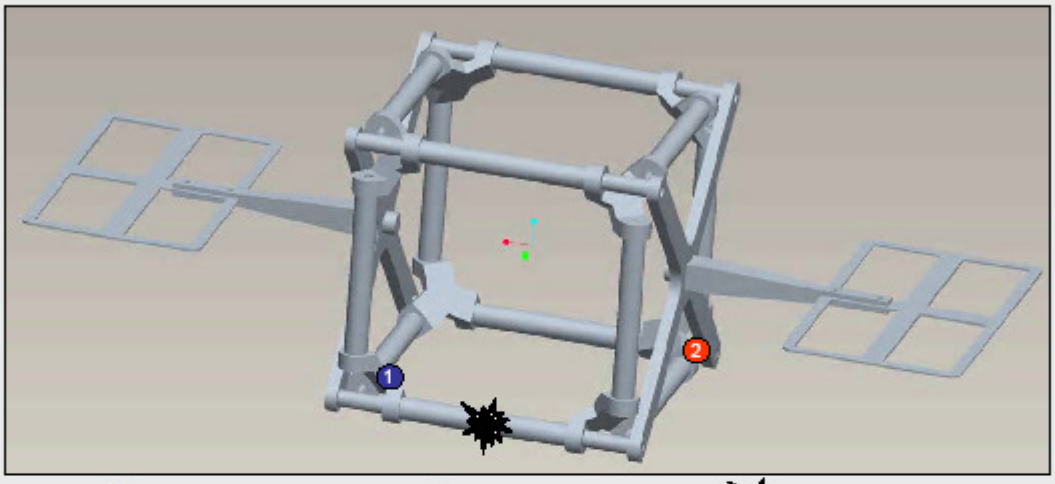
1 Accelerometer #1 Location 2 Accelerometer #2 Location Strike Location



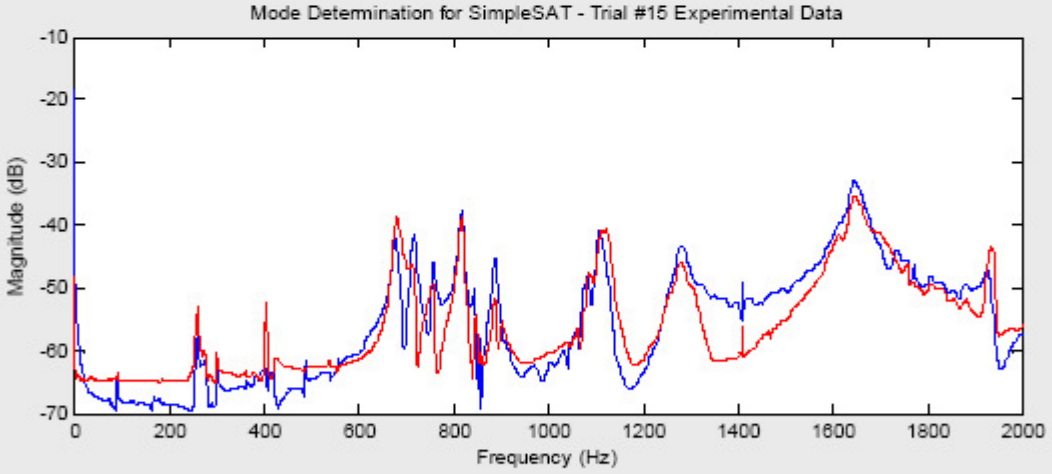
Mode Frequency Analysis										
Accelerometer	1st Mode		2nd Mode		3rd Mode		4th Mode		5th Mode	
	dB	Hz	dB	Hz	dB	Hz	dB	Hz	dB	Hz
1	-61.52	90	-56.14	260	-54.62	405	-38.7	680	-40.08	817.5
2	-62.44	90	-56.76	277.5	-50.88	405	-37.11	680	-38.09	817.5

SimpleSat Vibrational Analysis

Trial #15



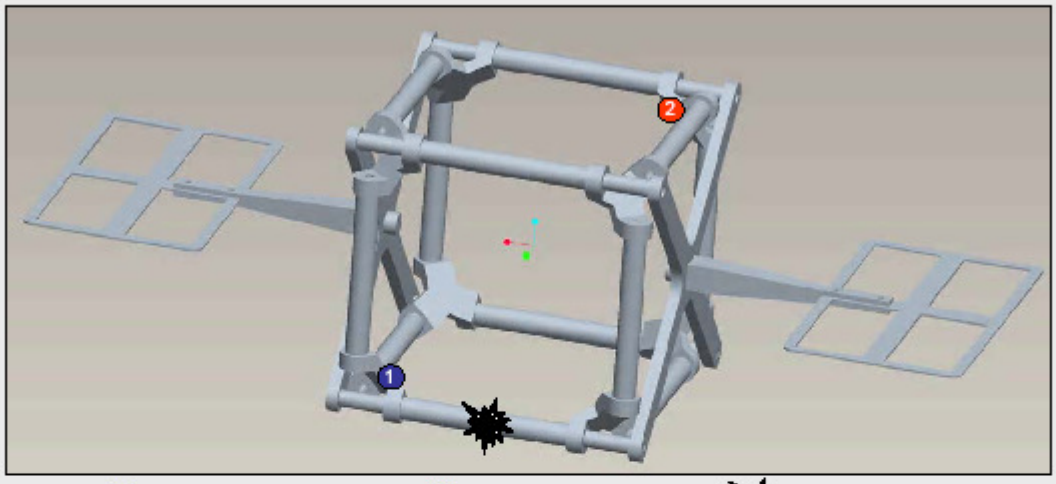
1 Accelerometer #1 Location 2 Accelerometer #2 Location Strike Location



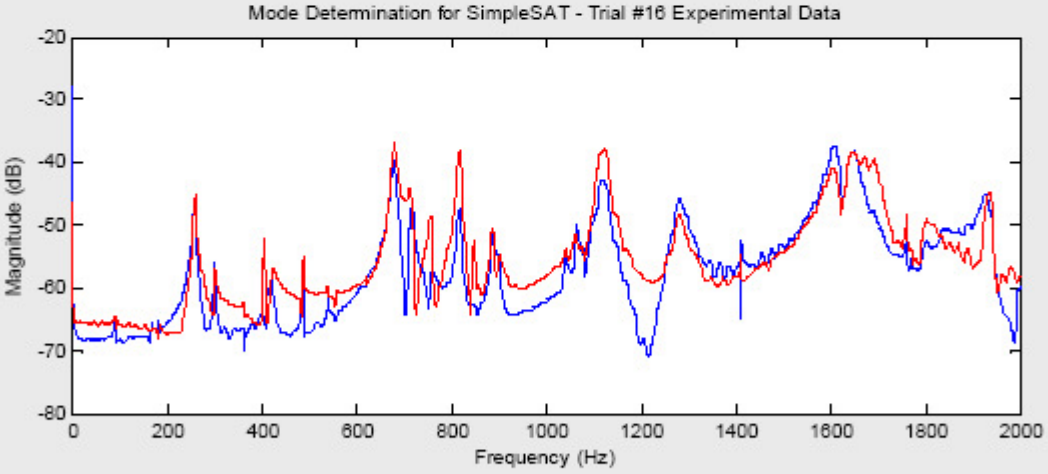
Mode Frequency Analysis										
Accelerometer	1st Mode		2nd Mode		3rd Mode		4th Mode		5th Mode	
	dB	Hz	dB	Hz	dB	Hz	dB	Hz	dB	Hz
1	-63.58	90	-53.12	260	-52.31	405	-38.52	680	-38.63	817.5
2	-63.65	90	-56.07	260	-62.83	405	-42.03	680	-37.62	817.5

SimpleSat Vibrational Analysis

Trial #16



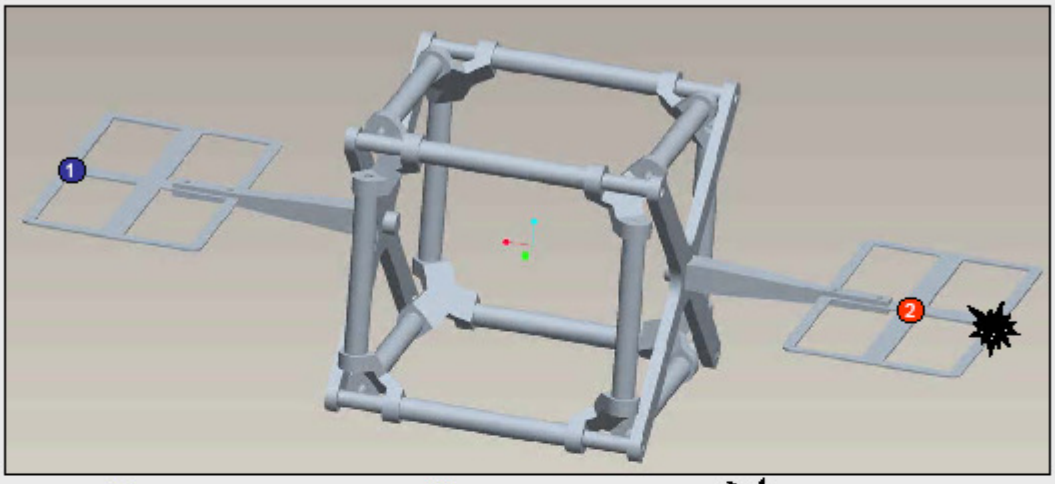
1 Accelerometer #1 Location 2 Accelerometer #2 Location Strike Location



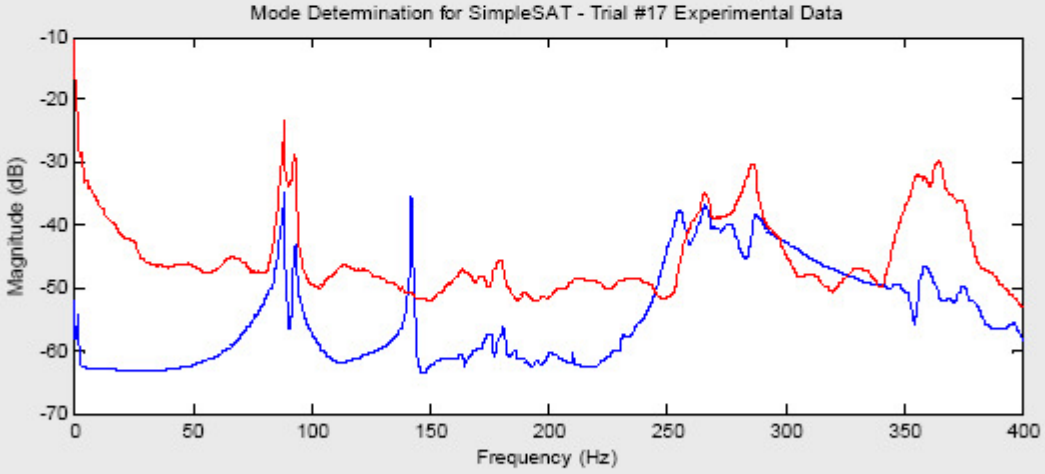
Mode Frequency Analysis										
Accelerometer	1st Mode		2nd Mode		3rd Mode		4th Mode		5th Mode	
	dB	Hz	dB	Hz	dB	Hz	dB	Hz	dB	Hz
1	-65.17	90	-46.24	260	-58.67	420	-39.65	680	-47.52	817.5
2	-64.58	90	-45.34	260	-52.11	405	-36.93	680	-37.96	817.5

SimpleSat Vibrational Analysis

Trial #17



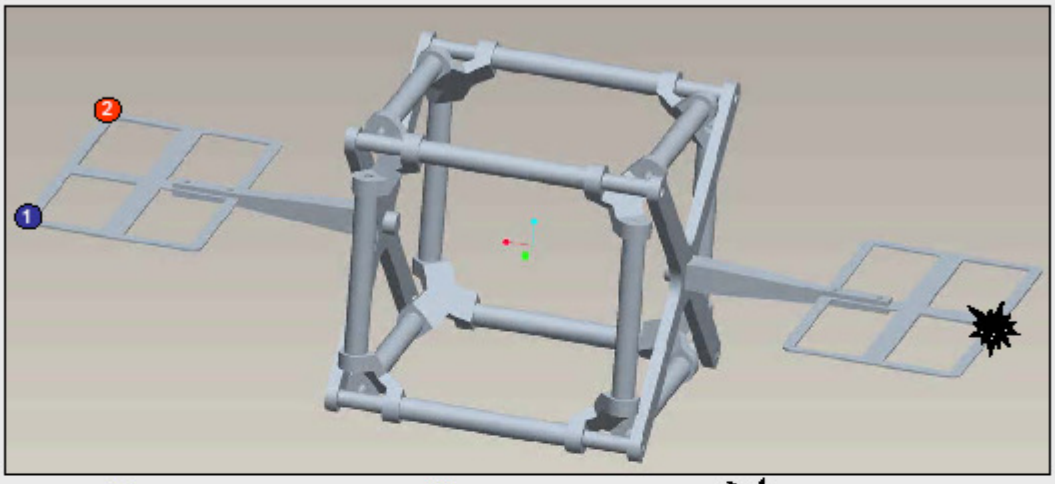
1 Accelerometer #1 Location 2 Accelerometer #2 Location Strike Location



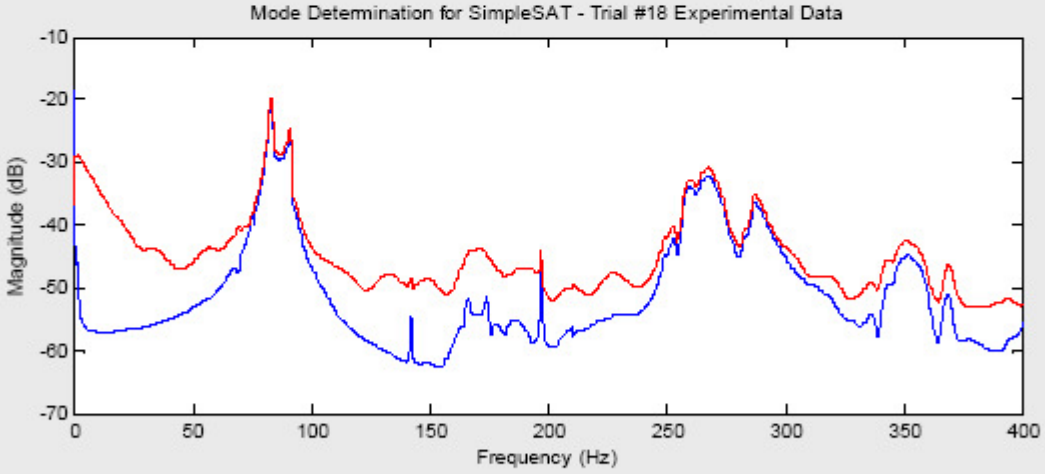
Mode Frequency Analysis										
Accelerometer	1st Mode		2nd Mode		3rd Mode		4th Mode		5th Mode	
	dB	Hz	dB	Hz	dB	Hz	dB	Hz	dB	Hz
1			-34.68	88.5	-42.89	93.5	-35.55	142.5	-56	181
2			-23.08	88.5	-28.73	93			-45.47	180

SimpleSat Vibrational Analysis

Trial #18



1 Accelerometer #1 Location 2 Accelerometer #2 Location Strike Location



Mode Frequency Analysis										
Accelerometer	1st Mode		2nd Mode		3rd Mode		4th Mode		5th Mode	
	dB	Hz	dB	Hz	dB	Hz	dB	Hz	dB	Hz
1			-20.6	83	-25.5	91	-54.55	142.5	-44.15	197
2			-19.74	83	-24.77	91	-48.48	142	-44.10	197

Appendix C: Finite Element Model Construction Code

Main.m

```
clear all;clf;clc;
Time(1,:)=clock;
ModelProps;
Build = 0
if Build == 0
    Beam;
    CornerCreate;
    CollarCreate;
    SolarCrossCreate;
    SolarSupportCreate;
    SolarPanelCreate;
    save Body.mat Body*;
    save Connect.mat Connect*;
    save Corner.mat Corner*;
    save Collar.mat Collar*;
    save SolarCross.mat SolarCross*;
    save SolarSupport.mat SolarSupport*;
    save SolarPanel.mat SolarPanel*;
    save L.mat L*;
    save N.mat N*;
elseif Build == 1
    load Body.mat
    load Connect.mat
    load Corner.mat
    load Collar.mat
    load SolarCross.mat
    load SolarSupport.mat
    load SolarPanel.mat
    load L.mat
    load N.mat
end
Time(2,:)=clock
ModelTree_new;
%ModelTreeCheck;
ModelConst;
Time(3,:)=clock
ModelBuild;
Spymatrix;
Time(4,:)=clock
```

ModelProps.m

```
model.pl = [1   0.411646966 1.04E07         0.3     0.000252    4.016E06;
            2   0.411646966 3.04579e+007    0.285   0.000729866 1.18513e+007];


typ=fe_mat('p_beam','IN',1); % standard beam
model.il = [1 typ  1.66667E-05  8.33333E-06 8.33333E-06 0.01            % .1 X .1
            2 typ  0.09817477   0.049087385 0.049087385 0.785398163     % .5 circle
            3 typ  6.77083E-05  2.60417E-06 6.51042E-05 0.0125          % .25 X .05
            4 typ  0.000526042  5.20833E-06 0.000520833 0.025           % .5 X .05
            5 typ  6.77083E-05  6.51042E-05 2.60417E-06 0.0125          % .05 X .25
            6 typ  0.000526042  0.000520833 5.20833E-06 0.025           % .05 X .5
            7 typ  0.000651042  0.000325521 0.000325521 0.0625          % .25 X .25
            8 typ  0.003295898  0.001647949 0.001647949 0.140625        % .375  .375
            9 typ  0.010416667  0.005208333 0.005208333 0.25            % .5 X .5
            10 typ 0.003255208  0.000651042 0.002604167 0.125           % .5 X .25
            11 typ 0.003255208  0.002604167 0.000651042 0.125];         % .25 X .5
```

Beam.M

```
close all;clf;

eval(demosdt('echooff'))

L_Text={'Four';'Six';'Eight'};
N_Text={'Low';'Med';'High'};
FEelt=[];

L=[4;6;8];
N=[10;100;1000];

for i=1:length(L);
    femesh('reset'); % initialize FEMESH
    for j=1:length(N);
        a=(L(i)-(.2))/(N(j)-3);    %note the 3, 2 nodes + 1 for start
        FEelt(1,:)=[Inf     abs('beam1')];

        for n=1:(N(j)-2);
             FEnode(n,:)= [(n)  0 0 0  (.1+((n-1)*a)) 0 0];
             FEelt((n+1),:)=[(n) (n+1) 1 2 50 0];
        end

        FEelt((n+1),:)=[(n) (n+2) 1 2 50 0];
        FEnode((N(j)-1),:) = [(N(j)-1)  0 0 0  0 0 0];
        FEelt(N(j),:)=[(N(j)-1) 1 1 2 50 0];
        FEnode(N(j),:) = [N(j)  0 0 0  L(i) 0 0];

        text_model=strcat('Body.',char(L_Text(i)),'.',char(N_Text(j)),'.')
        text_M=strcat(text_model,'model=femesh');
        eval(text_M)
%        model=femesh;

        text_Cnxs=strcat(text_model,'Cnxs=2');
        eval(text_Cnxs)

        quote=char(39);

        text_pl=strcat(text_model,'model.pl=model.pl');
        eval(text_pl)
%        model.pl=[m_elastic('dbval 1 aluminum')]
        text_il=strcat(text_model,'model.il=model.il');
        eval(text_il)
%        model.il=[p_beam('dbval 1 rectangle .1 .1')]

        text_mknaming=strcat('[Beam.M,Beam.K,Beam.mdof]=');
        text_modelselect=strcat('fe_mknl(',text_model,'model);');
        text_mknl=strcat(text_mknaming,text_modelselect);
        eval(text_mknl)
%
[Beam.Length(i).Res(j).M,Beam.Length(i).Res(j).K,Beam.Length(i).Res(j).mdof]=fe_mknl(model);

        H(i,j)=length(Beam.M);

        text_bodyM=strcat(text_model,'M=Beam.M(1:(H(i,j)-12),1:(H(i,j)-12));');
        eval(text_bodyM)
        text_bodyK=strcat(text_model,'K=Beam.K(1:(H(i,j)-12),1:(H(i,j)-12));');
        eval(text_bodyK)
        text_bodySize=strcat(text_model,'Size=length(',text_model,'M);');
        eval(text_bodySize)
    end
end

        Connect.Pos.M=Beam.M((H(i,j)-11):(H(i,j)-6),(H(i,j)-11):(H(i,j)-6));
        Connect.Pos.Mcx=Beam.M(1:6,(H(i,j)-11):(H(i,j)-6));
        Connect.Pos.K=Beam.K((H(i,j)-11):(H(i,j)-6),(H(i,j)-11):(H(i,j)-6));
        Connect.Pos.Kcx=Beam.K(1:6,(H(i,j)-11):(H(i,j)-6));
        Connect.Pos.Size=length(Connect.Pos.M);
```

```
        Connect.Neg.M=Beam.M((H(i,j)-5):H(i,j),(H(i,j)-5):H(i,j));
        Connect.Neg.Mcx=Beam.M((H(i,j)-17):(H(i,j)-12),(H(i,j)-5):H(i,j));
        Connect.Neg.K=Beam.K((H(i,j)-5):H(i,j),(H(i,j)-5):H(i,j));
        Connect.Neg.Kcx=Beam.K((H(i,j)-17):(H(i,j)-12),(H(i,j)-5):H(i,j));
        Connect.Neg.Size=length(Connect.Neg.M);
```

CornerCreate.m

```
eval(demosdt('echooff'))

femesh('reset'); % initialize FEMESH
FEelt=[];

FEnode = [1  0 0 0  1 0 0.5;
          2  0 0 0  1 0.5 0;
          3  0 0 0  0.5 1 0;
          4  0 0 0  0 1 0.5;
          5  0 0 0  0 0.5 1;
          6  0 0 0  0.5 0 1;
          7  0 0 0  1 0 0;
          8  0 0 0  0 1 0;
          9  0 0 0  0 0 1;
          10  0 0 0  1.1 0 0;
          11  0 0 0  0 1.1 0;
          12  0 0 0  0 0 1.1];

FEelt = [Inf     abs('beam1');
         1 2 1 7 50 0;
         2 3 1 7 50 0;
         3 4 1 7 50 0;
         4 5 1 7 50 0;
         5 6 1 7 50 0;
         6 1 1 7 50 0;
         1 7 1 7 50 0;
         7 2 1 7 50 0;
         3 8 1 7 50 0;
         8 4 1 7 50 0;
         5 9 1 7 50 0;
         9 6 1 7 50 0;
         7 10 1 7 50 0;
         8 11 1 7 50 0;
         9 12 1 7 50 0];
%         7 13 1 7 50 0];

Corner.model=femesh;
Corner.Cnxs=3;

Corner.model.pl=model.pl

Corner.model.il=model.il

[C.M,C.K,C.mdof]=fe_mknl(Corner.model);

lsize=length(C.M)%-(6*Corner.Cnxs)
Corner.M=C.M(1:lsize,1:lsize);
Corner.K=C.K(1:lsize,1:lsize);

Corner.Size = length(Corner.M);

cf=feplot; cf.model=Corner.model;
```

CollarCreate.m

```
close all;clf;

eval(demosdt('echooff'))

ClrL_Text={'One'};
ClrN_Text={'Low';'High'};
FEelt=[];

ClrL=[1.25];
ClrN=[10;100];

for i=1:length(ClrL);
    femesh('reset'); % initialize FEMESH
    for j=1:length(ClrN);
        a=(ClrL(i)-(.2))/(ClrN(j)-3);    %note the 3, 2 nodes + 1 for start
        FEelt(1,:)=[Inf     abs('beam1')];

        for n=1:(ClrN(j)-2);
             FEnode(n,:)= [(n)  0 0 0  (.1+((n-1)*a)) 0 0];
             FEelt((n+1),:)=[(n) (n+1) 1 2 50 0];
        end

        FEelt((n+1),:)=[(n) (n+2) 1 2 50 0];
        FEnode((ClrN(j)-1),:) = [(ClrN(j)-1)  0 0 0  0 0 0];
        FEelt(ClrN(j),:)=[(ClrN(j)-1) 1 1 2 50 0];
        FEnode(ClrN(j),:) = [ClrN(j)  0 0 0  ClrL(i) 0 0];

        text_model=strcat('Collar.',char(ClrN_Text(j)),'.')
        text_M=strcat(text_model,'model=femesh');
        eval(text_M)

        text_Cnxs=strcat(text_model,'Cnxs=2');
        eval(text_Cnxs)

        quote=char(39);

        text_pl=strcat(text_model,'model.pl=model.pl');
        eval(text_pl)

        text_il=strcat(text_model,'model.il=model.il');
        eval(text_il)

        text_mknaming=strcat('[Collar.M,Collar.K,Collar.mdof]=');
        text_modelselect=strcat('fe_mknl(',text_model,'model);');
        text_mknl=strcat(text_mknaming,text_modelselect);
        eval(text_mknl)

        H(i,j)=length(Collar.M);

        text_bodyM=strcat(text_model,'M=Collar.M(1:(H(i,j)-12),1:(H(i,j)-12));');
        eval(text_bodyM)
        text_bodyK=strcat(text_model,'K=Collar.K(1:(H(i,j)-12),1:(H(i,j)-12));');
        eval(text_bodyK)
        text_bodySize=strcat(text_model,'Size=length(',text_model,'M);');
        eval(text_bodySize)



    end
end
```

SolarCrossCreate.m

```
%clear all;close all;clf;

eval(demosdt('echooff'))

femesh('reset'); % initialize FEMESH
FEelt=[];

FEnode = [1  0 0 0  0 0 0;
          2  0 0 0  -1 -1 0;
          3  0 0 0  1 -1 0;
          4  0 0 0  1 1 0;
          5  0 0 0  -1 1 0;
          6  0 0 0  -2 -2 0;
          7  0 0 0  2 -2 0;
          8  0 0 0  2 2 0;
          9  0 0 0  -2 2 0;
          10 0 0 0  -3 -3 0;
          11 0 0 0  3 -3 0;
          12 0 0 0  3 3 0;
          13 0 0 0  -3 3 0;
          14 0 0 0  -3 -3 0.1;
          15 0 0 0  3 -3 0.1;
          16 0 0 0  3 3 0.1;
          17 0 0 0  -3 3 0.1];

%FEelt(1,:)=[Inf     abs('beam1')];
FEelt = [Inf     abs('beam1');
         1 2 1 11 50 0;
         2 6 1 11 50 0;
         6 10 1 11 50 0;
         1 3 1 11 50 0;
         3 7 1 11 50 0;
         7 11 1 11 50 0;
         1 4 1 11 50 0;
         4 8 1 11 50 0;
         8 12 1 11 50 0;
         1 5 1 11 50 0;
         5 9 1 11 50 0;
         9 13 1 11 50 0;
         10 14 1 11 50 0;
         11 15 1 11 50 0;
         12 16 1 11 50 0;
         13 17 1 11 50 0];

SolarCross.model=femesh;
SolarCross.Cnxs=4;

SolarCross.model.pl=model.pl

SolarCross.model.il=model.il

[SP.M,SP.K,SP.mdof]=fe_mknl(SolarCross.model);

lsize=length(SP.M)-(6*SolarCross.Cnxs)
SolarCross.M=SP.M(1:lsize,1:lsize);
SolarCross.K=SP.K(1:lsize,1:lsize);

SolarCross.Size = length(SolarCross.M);

cf=feplot; cf.model=SolarCross.model;
```

SolarSupportCreate.m

```
%clear all;close all;clf;

eval(demosdt('echooff'))

femesh('reset'); % initialize FEMESH
FEelt=[];

FEnode = [1  0 0 0  0 0 0;
          2  0 0 0  0 1 0;
          3  0 0 0  0 2 0;
          4  0 0 0  0 3 0;
          5  0 0 0  0 3.5 0;
          6  0 0 0  .2 3.5 0;
          7  0 0 0  -.2 3.5 0;
          8  0 0 0  .2 4 0;
          9 0 0 0  -.2 4 0;
          10  0 0 0  .2 5 0;
          11 0 0 0  -.2 5 0];

FEelt = [Inf     abs('beam1');
         1 2 1 9 50 0;
         2 3 1 9 50 0;
         3 4 1 8 50 0;
         4 5 1 8 50 0;
         5 6 1 8 50 0;
         6 8 1 7 50 0;
         8 10 1 7 50 0;
         5 7 1 8 50 0;
         7 9 1 7 50 0;
         9 11 1 7 50 0];

SolarSupport.model=femesh;
SolarSupport.Cnxs=0;

SolarSupport.model.pl=model.pl

SolarSupport.model.il=model.il

[SP.M,SP.K,SP.mdof]=fe_mknl(SolarSupport.model);

lsize=length(SP.M)-(6*SolarSupport.Cnxs)
SolarSupport.M=SP.M(1:lsize,1:lsize);
SolarSupport.K=SP.K(1:lsize,1:lsize);

SolarSupport.Size = length(SolarSupport.M);

cf=feplot; cf.model=SolarSupport.model;
```

SolarPanelCreate.m

```
%clear all;close all;clf;

eval(demosdt('echooff'))

%Low Resolution

femesh('reset'); % initialize FEMESH
FEelt=[];

FEnode = [1  0 0 0  0 0 0;          2  0 0 0  0 1 0;
          3  0 0 0  0 1.5 0;        4  0 0 0  0 2 0;
          5  0 0 0  0 3 0;          6  0 0 0  .5 3 0;
          7  0 0 0  1.5 3 0;        8  0 0 0  2.5 3 0;
```

```
          9  0 0 0  2.5 2 0;         10 0 0 0  2.5 1.5 0;
          11 0 0 0  2.5 1 0;         12 0 0 0  2.5 0 0;
          13 0 0 0  1.5 0 0;         14 0 0 0  0.5 0 0;
          15 0 0 0  .5 1.5 0;        16 0 0 0  1.5 1.5 0;
          17 0 0 0  -.5 3 0;         18 0 0 0  -1.5 3 0;
          19 0 0 0  -2.5 3 0;        20 0 0 0  -2.5 2 0;
          21 0 0 0  -2.5 1.5 0;      22 0 0 0  -2.5 1 0;
          23 0 0 0  -2.5 0 0;        24 0 0 0  -1.5 0 0;
          25 0 0 0  -.5 0 0;         26 0 0 0  -.5 1.5 0;
          27 0 0 0  -1.5 1.5 0];
%FEelt(1,:)=[Inf     abs('beam1')];
FEelt = [Inf     abs('beam1');
         1 2 1 4 50 0;          2 3 1 4 50 0;
         3 4 1 4 50 0;          4 5 1 4 50 0;
         5 6 1 3 50 0;          6 7 1 3 50 0;
         7 8 1 3 50 0;          8 9 1 3 50 0;
         9 10 1 3 50 0;         10 11 1 3 50 0;
         11 12 1 3 50 0;        12 13 1 3 50 0;
         13 14 1 3 50 0;        14 1 1 3 50 0;
         3 15 1 4 50 0;         15 16 1 4 50 0;
         16 10 1 4 50 0;        5 17 1 3 50 0;
         17 18 1 3 50 0;        18 19 1 3 50 0;
         19 20 1 3 50 0;        20 21 1 3 50 0;
         21 22 1 3 50 0;        22 23 1 3 50 0;
         23 24 1 3 50 0;        24 25 1 3 50 0;
         25 1 1 3 50 0;         3 26 1 4 50 0;
         26 27 1 4 50 0;        27 21 1 4 50 0];

SolarPanel.Low.model=femesh;
SolarPanel.Low.Cnxs=0;

SolarPanel.Low.model.pl=model.pl

SolarPanel.Low.model.il=model.il

[SP.M,SP.K,SP.mdof]=fe_mknl(SolarPanel.Low.model);

lsize=length(SP.M)-(6*SolarPanel.Low.Cnxs)
SolarPanel.Low.M=SP.M(1:lsize,1:lsize);
SolarPanel.Low.K=SP.K(1:lsize,1:lsize);

SolarPanel.Low.Size = length(SolarPanel.Low.M);



%Medium Resolution
femesh('reset'); % initialize FEMESH
FEelt=[];

FEnode = [1  0 0 0  0 0 0;          2  0 0 0  0 1 0;
          3  0 0 0  0 1.5 0;        4  0 0 0  0 2 0;
          5  0 0 0  0 3 0;          6  0 0 0  .5 3 0;
          7  0 0 0  1.5 3 0;        8  0 0 0  2.5 3 0;
          9  0 0 0  2.5 2 0;        10 0 0 0  2.5 1.5 0;
          11 0 0 0  2.5 1 0;        12 0 0 0  2.5 0 0;
          13 0 0 0  1.5 0 0;        14 0 0 0  0.5 0 0;
          15 0 0 0  .5 1.5 0;       16 0 0 0  1.5 1.5 0;
          17 0 0 0  -.5 3 0;        18 0 0 0  -1.5 3 0;
          19 0 0 0  -2.5 3 0;       20 0 0 0  -2.5 2 0;
          21 0 0 0  -2.5 1.5 0;     22 0 0 0  -2.5 1 0;
          23 0 0 0  -2.5 0 0;       24 0 0 0  -1.5 0 0;
          25 0 0 0  -.5 0 0;        26 0 0 0  -.5 1.5 0;
          27 0 0 0  -1.5 1.5 0;     28 0 0 0  0 .5 0;
          29 0 0 0  0 2.5 0;        30 0 0 0  1 3 0;
          31 0 0 0  2 3 0;          32 0 0 0  2.5 2.5 0;
          33 0 0 0  2.5 0.5 0;      34 0 0 0  2 0 0;
          35 0 0 0  1 0 0;          36 0 0 0  -1 3 0;
          37 0 0 0  -2 3 0;         38 0 0 0  -2.5 2.5 0;
```

```
          39 0 0 0  -2.5 .5 0;      40 0 0 0  -2 0 0;
          41 0 0 0  -1 0 0;         42 0 0 0  -1 1.5 0;
          43 0 0 0  -2 1.5 0;       44 0 0 0  1 1.5 0;
          45 0 0 0  2 1.5 0];


%FEelt(1,:)=[Inf     abs('beam1')];
FEelt = [Inf     abs('beam1');

          2 3 1 6 50 0;
          3 4 1 6 50 0;
          5 6 1 5 50 0;
          9 10 1 5 50 0;        10 11 1 5 50 0;
          14 1 1 5 50 0;
          3 15 1 6 50 0;
          5 17 1 5 50 0;
          20 21 1 5 50 0;
          21 22 1 5 50 0;
          25 1 1 5 50 0;        3 26 1 6 50 0;
          1 28 1 6 50 0;        28 2 1 6 50 0;
          4 29 1 6 50 0;        29 5 1 6 50 0;
          6 30 1 5 50 0;         30 7 1 5 50 0;
          7 31 1 5 50 0;         31 8 1 5 50 0;
          8 32 1 5 50 0;         32 9 1 5 50 0;
          11 33 1 5 50 0;        33 12 1 5 50 0;
          12 34 1 5 50 0;        34 13 1 5 50 0;
          13 35 1 5 50 0;        35 14 1 5 50 0;
          17 36 1 5 50 0;        36 18 1 5 50 0;
          18 37 1 5 50 0;        37 19 1 5 50 0;
          19 38 1 5 50 0;        38 20 1 5 50 0;
          22 39 1 5 50 0;        39 23 1 5 50 0;
          23 40 1 5 50 0;        40 24 1 5 50 0;
          24 41 1 5 50 0;        41 25 1 5 50 0;
          26 42 1 6 50 0;        42 27 1 6 50 0;
          27 43 1 6 50 0;        43 21 1 6 50 0;
          15 44 1 6 50 0;        44 16 1 6 50 0;
          16 45 1 6 50 0;        45 10 1 6 50 0];

SolarPanel.Med.model=femesh;
SolarPanel.Med.Cnxs=0;

SolarPanel.Med.model.pl=model.pl

SolarPanel.Med.model.il=model.il

[SP.M,SP.K,SP.mdof]=fe_mknl(SolarPanel.Med.model);

lsize=length(SP.M)-(6*SolarPanel.Med.Cnxs)
SolarPanel.Med.M=SP.M(1:lsize,1:lsize);
SolarPanel.Med.K=SP.K(1:lsize,1:lsize);

SolarPanel.Med.Size = length(SolarPanel.Med.M);

cf=feplot; cf.model=SolarPanel.Med.model;%cf.def=def;
```

ModelTree_new.m

```
Tree.Mbr(1).Type='Body.Six.Low';%1,2,3
Tree.Mbr(1).Loc=[1,0,0];
Tree.Mbr(1).Vec=[1,0,0];
Tree.Mbr(1).Rot=[0];
Tree.Mbr(1).CntMbr=0;

Tree.Mbr(2).Type='Body.Four.Low';%4,5,6
```

```
Tree.Mbr(2).Loc=[L(2)+2,1,0];
Tree.Mbr(2).Vec=[0,1,0];
Tree.Mbr(2).Rot=[0];
Tree.Mbr(2).CntMbr=0;

Tree.Mbr(3).Type='Body.Six.Low';%7,8,9
Tree.Mbr(3).Loc=[1,L(1)+2,0];
Tree.Mbr(3).Vec=[1,0,0];
Tree.Mbr(3).Rot=[0];
Tree.Mbr(3).CntMbr=0;

Tree.Mbr(4).Type='Body.Four.Low';%10,11,12
Tree.Mbr(4).Loc=[0,1,0];
Tree.Mbr(4).Vec=[0,1,0];
Tree.Mbr(4).Rot=[0];
Tree.Mbr(4).CntMbr=0;

Tree.Mbr(5).Type='Body.Four.Low';%13,14,15
Tree.Mbr(5).Loc=[0,0,1];
Tree.Mbr(5).Vec=[0,0,1];
Tree.Mbr(5).Rot=[0];
Tree.Mbr(5).CntMbr=0;

Tree.Mbr(6).Type='Body.Four.Low';%16,17,18
Tree.Mbr(6).Loc=[L(2)+2,0,1];
Tree.Mbr(6).Vec=[0,0,1];
Tree.Mbr(6).Rot=[0];
Tree.Mbr(6).CntMbr=0;

Tree.Mbr(7).Type='Body.Four.Low';%19,20,21
Tree.Mbr(7).Loc=[L(2)+2,L(1)+2,1];
Tree.Mbr(7).Vec=[0,0,1];
Tree.Mbr(7).Rot=[0];
Tree.Mbr(7).CntMbr=0;

Tree.Mbr(8).Type='Body.Four.Low';%22,23,24
Tree.Mbr(8).Loc=[0,L(1)+2,1];
Tree.Mbr(8).Vec=[0,0,1];
Tree.Mbr(8).Rot=[0];
Tree.Mbr(8).CntMbr=0;

Tree.Mbr(9).Type='Body.Six.Low';%25,26,27
Tree.Mbr(9).Loc=[1,0,L(1)+2];
Tree.Mbr(9).Vec=[1,0,0];
Tree.Mbr(9).Rot=[0];
Tree.Mbr(9).CntMbr=0;

Tree.Mbr(10).Type='Body.Four.Low';%28,29,30
Tree.Mbr(10).Loc=[L(2)+2,1,L(1)+2];
Tree.Mbr(10).Vec=[0,1,0];
Tree.Mbr(10).Rot=[0];
Tree.Mbr(10).CntMbr=0;

Tree.Mbr(11).Type='Body.Six.Low';%31,32,33
Tree.Mbr(11).Loc=[1,L(1)+2,L(1)+2];
Tree.Mbr(11).Vec=[1,0,0];
Tree.Mbr(11).Rot=[0];
Tree.Mbr(11).CntMbr=0;

Tree.Mbr(12).Type='Body.Four.Low';%34,35,36
Tree.Mbr(12).Loc=[0,1,L(1)+2];
Tree.Mbr(12).Vec=[0,1,0];
Tree.Mbr(12).Rot=[0];
Tree.Mbr(12).CntMbr=0;


Tree.Mbr(13).Type='Corner';%1
Tree.Mbr(13).Loc=[0,0,0];
Tree.Mbr(13).Vec=[1,0,0];
```

```
Tree.Mbr(13).Rot=[0];
Tree.Mbr(13).CntMbr=1;
Tree.Mbr(13).Cnx(1).Mbr=1;
Tree.Mbr(13).Cnx(1).Type='Connect.Pos';
Tree.Mbr(13).Cnx(2).Mbr=4;
Tree.Mbr(13).Cnx(2).Type='Connect.Pos';
Tree.Mbr(13).Cnx(3).Mbr=5;
Tree.Mbr(13).Cnx(3).Type='Connect.Pos';
%Tree.Mbr(13).Cnx(4).Mbr=0;

Tree.Mbr(14).Type='Corner';%2
Tree.Mbr(14).Loc=[L(2)+2,0,0];
Tree.Mbr(14).Vec=[0,0,1];
Tree.Mbr(14).Rot=[0];
Tree.Mbr(14).CntMbr=1;
Tree.Mbr(14).Cnx(1).Mbr=6;
Tree.Mbr(14).Cnx(1).Type='Connect.Pos';
Tree.Mbr(14).Cnx(2).Mbr=2;
Tree.Mbr(14).Cnx(2).Type='Connect.Pos';
Tree.Mbr(14).Cnx(3).Mbr=1;
Tree.Mbr(14).Cnx(3).Type='Connect.Neg';

Tree.Mbr(15).Type='Corner';%3
Tree.Mbr(15).Loc=[L(2)+2,L(1)+2,0];
Tree.Mbr(15).Vec=[-1,0,0];
Tree.Mbr(15).Rot=[pi];
Tree.Mbr(15).CntMbr=1;
Tree.Mbr(15).Cnx(1).Mbr=3;
Tree.Mbr(15).Cnx(1).Type='Connect.Neg';
Tree.Mbr(15).Cnx(2).Mbr=2;
Tree.Mbr(15).Cnx(2).Type='Connect.Neg';
Tree.Mbr(15).Cnx(3).Mbr=7;
Tree.Mbr(15).Cnx(3).Type='Connect.Pos';

Tree.Mbr(16).Type='Corner';%4
Tree.Mbr(16).Loc=[0,L(1)+2,0];
Tree.Mbr(16).Vec=[0,-1,0];
Tree.Mbr(16).Rot=[0];
Tree.Mbr(16).CntMbr=1;
Tree.Mbr(16).Cnx(1).Mbr=4;
Tree.Mbr(16).Cnx(1).Type='Connect.Neg';
Tree.Mbr(16).Cnx(2).Mbr=3;
Tree.Mbr(16).Cnx(2).Type='Connect.Pos';
Tree.Mbr(16).Cnx(3).Mbr=8;
Tree.Mbr(16).Cnx(3).Type='Connect.Pos';

Tree.Mbr(17).Type='Corner';%5
Tree.Mbr(17).Loc=[0,0,L(1)+2];
Tree.Mbr(17).Vec=[0,0,-1];
Tree.Mbr(17).Rot=[0];
Tree.Mbr(17).CntMbr=1;
Tree.Mbr(17).Cnx(1).Mbr=5;
Tree.Mbr(17).Cnx(1).Type='Connect.Neg';
Tree.Mbr(17).Cnx(2).Mbr=12;
Tree.Mbr(17).Cnx(2).Type='Connect.Pos';
Tree.Mbr(17).Cnx(3).Mbr=9;
Tree.Mbr(17).Cnx(3).Type='Connect.Pos';

Tree.Mbr(18).Type='Corner';%6
Tree.Mbr(18).Loc=[L(2)+2,0,L(1)+2];
Tree.Mbr(18).Vec=[-1,0,0];
Tree.Mbr(18).Rot=[0];
Tree.Mbr(18).CntMbr=1;
Tree.Mbr(18).Cnx(1).Mbr=9;
Tree.Mbr(18).Cnx(1).Type='Connect.Neg';
Tree.Mbr(18).Cnx(2).Mbr=10;
Tree.Mbr(18).Cnx(2).Type='Connect.Pos';
Tree.Mbr(18).Cnx(3).Mbr=6;
Tree.Mbr(18).Cnx(3).Type='Connect.Neg';
```

```
Tree.Mbr(19).Type='Corner';%7
Tree.Mbr(19).Loc=[L(2)+2,L(1)+2,L(1)+2];
Tree.Mbr(19).Vec=[-1,0,0];
Tree.Mbr(19).Rot=[-pi/2];
Tree.Mbr(19).CntMbr=1;
Tree.Mbr(19).Cnx(1).Mbr=11;
Tree.Mbr(19).Cnx(1).Type='Connect.Neg';
Tree.Mbr(19).Cnx(2).Mbr=7;
Tree.Mbr(19).Cnx(2).Type='Connect.Neg';
Tree.Mbr(19).Cnx(3).Mbr=10;
Tree.Mbr(19).Cnx(3).Type='Connect.Neg';

Tree.Mbr(20).Type='Corner';%8
Tree.Mbr(20).Loc=[0,L(1)+2,L(1)+2];
Tree.Mbr(20).Vec=[1,0,0];
Tree.Mbr(20).Rot=[pi];
Tree.Mbr(20).CntMbr=1;
Tree.Mbr(20).Cnx(1).Mbr=11;
Tree.Mbr(20).Cnx(1).Type='Connect.Pos';
Tree.Mbr(20).Cnx(2).Mbr=12;
Tree.Mbr(20).Cnx(2).Type='Connect.Neg';
Tree.Mbr(20).Cnx(3).Mbr=8;
Tree.Mbr(20).Cnx(3).Type='Connect.Neg';

Tree.Mbr(21).Type='Collar.Low';
Tree.Mbr(21).Loc=[1,0,0];
Tree.Mbr(21).Vec=[-1,0,0];
Tree.Mbr(21).Rot=[0];
Tree.Mbr(21).CntMbr=2;
Tree.Mbr(21).Cnx(1).Mbr=13;
Tree.Mbr(21).Cnx(1).Type='Connect.Pos';
Tree.Mbr(21).Cnx(1).Connecter='Cnx(1)';

Tree.Mbr(22).Type='Collar.Low';
Tree.Mbr(22).Loc=[L(2)+1,0,0];
Tree.Mbr(22).Vec=[1,0,0];
Tree.Mbr(22).Rot=[0];
Tree.Mbr(22).CntMbr=2;
Tree.Mbr(22).Cnx(1).Mbr=14;
Tree.Mbr(22).Cnx(1).Type='Connect.Pos';
Tree.Mbr(22).Cnx(1).Connecter='Cnx(3)';

Tree.Mbr(23).Type='Collar.Low';
Tree.Mbr(23).Loc=[L(2)+1,L(1)+2,0];
Tree.Mbr(23).Vec=[1,0,0];
Tree.Mbr(23).Rot=[0];
Tree.Mbr(23).CntMbr=2;
Tree.Mbr(23).Cnx(1).Mbr=15;
Tree.Mbr(23).Cnx(1).Type='Connect.Pos';
Tree.Mbr(23).Cnx(1).Connecter='Cnx(1)';

Tree.Mbr(24).Type='Collar.Low';
Tree.Mbr(24).Loc=[1,L(1)+2,0];
Tree.Mbr(24).Vec=[-1,0,0];
Tree.Mbr(24).Rot=[0];
Tree.Mbr(24).CntMbr=2;
Tree.Mbr(24).Cnx(1).Mbr=16;
Tree.Mbr(24).Cnx(1).Type='Connect.Pos';
Tree.Mbr(24).Cnx(1).Connecter='Cnx(2)';

Tree.Mbr(25).Type='Collar.Low';
Tree.Mbr(25).Loc=[1,0,L(1)+2];
Tree.Mbr(25).Vec=[-1,0,0];
Tree.Mbr(25).Rot=[0];
Tree.Mbr(25).CntMbr=2;
Tree.Mbr(25).Cnx(1).Mbr=17;
Tree.Mbr(25).Cnx(1).Type='Connect.Pos';
Tree.Mbr(25).Cnx(1).Connecter='Cnx(3)';
```

```
Tree.Mbr(26).Type='Collar.Low';
Tree.Mbr(26).Loc=[L(2)+1,0,L(1)+2];
Tree.Mbr(26).Vec=[1,0,0];
Tree.Mbr(26).Rot=[0];
Tree.Mbr(26).CntMbr=2;
Tree.Mbr(26).Cnx(1).Mbr=18;
Tree.Mbr(26).Cnx(1).Type='Connect.Pos';
Tree.Mbr(26).Cnx(1).Connecter='Cnx(1)';

Tree.Mbr(27).Type='Collar.Low';
Tree.Mbr(27).Loc=[L(2)+1,L(1)+2,L(1)+2];
Tree.Mbr(27).Vec=[1,0,0];
Tree.Mbr(27).Rot=[0];
Tree.Mbr(27).CntMbr=2;
Tree.Mbr(27).Cnx(1).Mbr=19;
Tree.Mbr(27).Cnx(1).Type='Connect.Pos';
Tree.Mbr(27).Cnx(1).Connecter='Cnx(1)';

Tree.Mbr(28).Type='Collar.Low';
Tree.Mbr(28).Loc=[1,L(1)+2,L(1)+2];
Tree.Mbr(28).Vec=[-1,0,0];
Tree.Mbr(28).Rot=[0];
Tree.Mbr(28).CntMbr=2;
Tree.Mbr(28).Cnx(1).Mbr=20;
Tree.Mbr(28).Cnx(1).Type='Connect.Pos';
Tree.Mbr(28).Cnx(1).Connecter='Cnx(1)';

Tree.Mbr(29).Type='SolarCross';%1
Tree.Mbr(29).Loc=[-.45,3,3];
Tree.Mbr(29).Vec=[0,0,-1];
Tree.Mbr(29).Rot=[0];
Tree.Mbr(29).CntMbr=1;
Tree.Mbr(29).Cnx(1).Mbr=25;
Tree.Mbr(29).Cnx(1).Type='Connect.Neg';
Tree.Mbr(29).Cnx(2).Mbr=21;
Tree.Mbr(29).Cnx(2).Type='Connect.Neg';
Tree.Mbr(29).Cnx(3).Mbr=24;
Tree.Mbr(29).Cnx(3).Type='Connect.Neg';
Tree.Mbr(29).Cnx(4).Mbr=28;
Tree.Mbr(29).Cnx(4).Type='Connect.Neg';

Tree.Mbr(30).Type='SolarCross';%1
Tree.Mbr(30).Loc=[L(2)+2.45,3,3];
Tree.Mbr(30).Vec=[0,0,1];
Tree.Mbr(30).Rot=[0];
Tree.Mbr(30).CntMbr=1;
Tree.Mbr(30).Cnx(1).Mbr=22;
Tree.Mbr(30).Cnx(1).Type='Connect.Neg';
Tree.Mbr(30).Cnx(2).Mbr=26;
Tree.Mbr(30).Cnx(2).Type='Connect.Neg';
Tree.Mbr(30).Cnx(3).Mbr=27;
Tree.Mbr(30).Cnx(3).Type='Connect.Neg';
Tree.Mbr(30).Cnx(4).Mbr=23;
Tree.Mbr(30).Cnx(4).Type='Connect.Neg';

Tree.Mbr(31).Type='SolarSupport';
Tree.Mbr(31).Loc=[-.65,3,3];
Tree.Mbr(31).Vec=[0,1,0];
Tree.Mbr(31).Rot=[pi/2];
Tree.Mbr(31).CntMbr=2;
Tree.Mbr(31).Cnx(1).Mbr=29;
Tree.Mbr(31).Cnx(1).Type='Connect.Pos';
Tree.Mbr(31).Cnx(1).Connecter='Pos';

Tree.Mbr(32).Type='SolarSupport';
Tree.Mbr(32).Loc=[L(2)+2.65,3,3];
Tree.Mbr(32).Vec=[0,-1,0];
Tree.Mbr(32).Rot=[pi/2];
```

```
Tree.Mbr(32).CntMbr=2;
Tree.Mbr(32).Cnx(1).Mbr=30;
Tree.Mbr(32).Cnx(1).Type='Connect.Pos';
Tree.Mbr(32).Cnx(1).Connecter='Pos';

Tree.Mbr(33).Type='SolarPanel.Med';
Tree.Mbr(33).Loc=[-4.65,3,3];
Tree.Mbr(33).Vec=[0,1,0];
Tree.Mbr(33).Rot=[0];
Tree.Mbr(33).CntMbr=3;
Tree.Mbr(33).Cnx(1).Mbr=31;
Tree.Mbr(33).Cnx(1).Type='Connect.Pos';
Tree.Mbr(33).Cnx(1).Connecter='Neg - 3';
Tree.Mbr(33).Cnx(2).Mbr=31;
Tree.Mbr(33).Cnx(2).Type='Connect.Neg';
Tree.Mbr(33).Cnx(2).Connecter='Neg - 2';
Tree.Mbr(33).Cnx(3).Mbr=31;
Tree.Mbr(33).Cnx(3).Type='Connect.Pos';
Tree.Mbr(33).Cnx(3).Connecter='Neg -1';
Tree.Mbr(33).Cnx(4).Mbr=31;
Tree.Mbr(33).Cnx(4).Type='Connect.Neg';
Tree.Mbr(33).Cnx(4).Connecter='Neg';

Tree.Mbr(34).Type='SolarPanel.Med';
Tree.Mbr(34).Loc=[L(2)+6.65,3,3];
Tree.Mbr(34).Vec=[0,-1,0];
Tree.Mbr(34).Rot=[0];
Tree.Mbr(34).CntMbr=3;
Tree.Mbr(34).Cnx(1).Mbr=32;
Tree.Mbr(34).Cnx(1).Type='Connect.Pos';
Tree.Mbr(34).Cnx(1).Connecter='Neg - 3';
Tree.Mbr(34).Cnx(2).Mbr=32;
Tree.Mbr(34).Cnx(2).Type='Connect.Neg';
Tree.Mbr(34).Cnx(2).Connecter='Neg - 2';
Tree.Mbr(34).Cnx(3).Mbr=32;
Tree.Mbr(34).Cnx(3).Type='Connect.Pos';
Tree.Mbr(34).Cnx(3).Connecter='Neg -1';
Tree.Mbr(34).Cnx(4).Mbr=32;
Tree.Mbr(34).Cnx(4).Type='Connect.Neg';
Tree.Mbr(34).Cnx(4).Connecter='Neg';
```

ModelConst.m

```
Model.pl=model.pl;
Model.il=model.il;
NC = 0;%Node Count
EC = 1;%Element Counter
fin=0;
for i=1:length(Tree.Mbr)
    i;
    t=strcat(char(Tree.Mbr(i).Type));
    t1=strcat(t,'.model.Node');
    t2=strcat(t,'.model.Elt');
    t3=strcat(t,'.Cnxs');
    h=eval(t2);
    g=eval(t3);
    f=eval(t1);
    Tree.Mbr(i).Node.Pos=NC+1;
    n=Tree.Mbr(i).Node.Pos;
    %Node Wright
    for j=1:(size(f)-g)
        j;
        NC=NC+1;
        Model.Node(NC,1:4)=[NC 0 0 0];
        v=strcat('v=',t1,'(j,5:7);');
```

```
        eval(v);
        [vprime,theta]=unitvec(Tree.Mbr(i).Vec,v,Tree.Mbr(i).Loc,Tree.Mbr(i).Rot);
        Tree.Mbr(i).Theta=theta;
        Model.Node(NC,5:7)=vprime;

    end
    Tree.Mbr(i).Node.Neg=NC;
   %ELT Wright
    Model.Elt(EC,:)=h(1,:);
    EC=EC+1;
    for m=1:(size(eval(t2))-(g+1))
        Model.Elt((EC),:)=[(h(m+1,1)+n-1) (h(m+1,2)+(n-1)) h((m+1),(3:6))];
        EC=EC+1;
    end

    if Tree.Mbr(i).CntMbr == 1
    for u=1:g
        Tree.Mbr(i).Node.Cnx(u) = Tree.Mbr(i).Node.Neg - g + u;

    end
    end

    if Tree.Mbr(i).CntMbr == 2

        Tree.Mbr(i).Node.Cnx = Tree.Mbr(i).Node.Pos;
    end
    if Tree.Mbr(i).CntMbr == 3

        Tree.Mbr(i).Node.Cnx(1) = Tree.Mbr(i).Node.Pos;
        Tree.Mbr(i).Node.Cnx(2) = Tree.Mbr(i).Node.Pos;
        Tree.Mbr(i).Node.Cnx(3) = Tree.Mbr(i).Node.Pos + 1;
        Tree.Mbr(i).Node.Cnx(4) = Tree.Mbr(i).Node.Pos + 1;
    end

    start=fin+1;
    temp=eval(Tree.Mbr(i).Type,'Size');
    fin=fin+temp.Size;
    [RotMat.M]=MatrixRotate(Tree.Mbr(i).Vec,temp.M,Tree.Mbr(i).Rot);
    Cube.M(start:fin,start:fin)=RotMat.M;
    [RotMat.K]=MatrixRotate(Tree.Mbr(i).Vec,temp.K,Tree.Mbr(i).Rot);
    Cube.K(start:fin,start:fin)=RotMat.K;
    Tree.Mbr(i).start=start;
    Tree.Mbr(i).fin=fin;

    if Tree.Mbr(i).CntMbr >= 1
        CnxNum=length(Tree.Mbr(i).Cnx);
        for p=1:CnxNum
            s=CnxNum*6-1;%initial cnx start point
            Tree.Mbr(i).Cnx(p).start=fin-s+(p-1)*6;
        end
    end



end

for i=1:length(Tree.Mbr)
    if Tree.Mbr(i).CntMbr >= 1
        for j=1:length(Tree.Mbr(i).Cnx)
            i;
            j;
            NC=NC+1;
            Model.Node(NC,1:4)=[NC 0 0 0];
            MN=Tree.Mbr(i).Cnx(j).Mbr ;  %Member Number

            len=length(Tree.Mbr(i).Cnx(1).Type);
            BS=Tree.Mbr(i).Cnx(j).Type(len-2:len); % Beam Side
            if Tree.Mbr(i).CntMbr == 1
                temp=strcat('BN=Tree.Mbr(MN).Node.',char(BS));  %  Beam Node
```

```
else if Tree.Mbr(i).CntMbr >= 2
    BScnx=Tree.Mbr(i).Cnx(j).Connecter; % Beam Side
    temp=strcat('BN=Tree.Mbr(MN).Node.',char(BScnx));  %  Beam Node
    end
end



eval(temp)
CN=Tree.Mbr(i).Node.Cnx(j);   %Connector Node
U=Model.Node(BN,5:7); V=Model.Node(CN,5:7); W=(U-V)/2+V;
Model.Node(NC,5:7)=W;
EC=length(Model.Elt);
EC=EC+1;
Model.Elt(EC,:)=[NC CN 2 8 50 0];
Model.Elt((EC+1),:)=[NC BN 2 8 50 0];

start=fin+1;
fin=start+size(Connect.Pos.M)-1;
rot = 0;
PreRotate.M=Connect.Pos.M+Connect.Neg.M;
[RotMat.M]=MatrixRotate(Tree.Mbr(MN).Vec,PreRotate.M,rot);
Cube.M(start:fin,start:fin)=RotMat.M;
PreRotate.K=Connect.Pos.K+Connect.Neg.K;
[RotMat.K]=MatrixRotate(Tree.Mbr(MN).Vec,PreRotate.K,rot);
Cube.K(start:fin,start:fin)=RotMat.K;

if BS == 'Pos'
    %beam
    nstart = Tree.Mbr(MN).start;
    nfin = nstart+size(Connect.Pos.Mcx)-1;
    PreRotate.M=Connect.Pos.Mcx;
    [RotMat.M]=MatrixRotate(Tree.Mbr(MN).Vec,PreRotate.M,rot);
    Cube.M(nstart:nfin,start:fin)=RotMat.M;
    PreRotate.K=Connect.Pos.Kcx;
    [RotMat.K]=MatrixRotate(Tree.Mbr(MN).Vec,PreRotate.K,rot);
    Cube.K(nstart:nfin,start:fin)=RotMat.K;
    %corner
    mstart = Tree.Mbr(i).Cnx(j).start;
    mfin = mstart+size(Connect.Neg.Mcx)-1;
    PreRotate.M=Connect.Neg.Mcx;
    [RotMat.M]=MatrixRotate(Tree.Mbr(MN).Vec,PreRotate.M,rot);
    Cube.M(mstart:mfin,start:fin)=RotMat.M;
    PreRotate.K=Connect.Neg.Kcx;
    [RotMat.K]=MatrixRotate(Tree.Mbr(MN).Vec,PreRotate.K,rot);
    Cube.K(mstart:mfin,start:fin)=RotMat.K;


else if BS == 'Neg'
        %beam
        nstart = Tree.Mbr(MN).fin - size(Connect.Neg.Mcx) + 1;
        nfin = nstart+size(Connect.Neg.Mcx)-1;
        PreRotate.M=Connect.Neg.Mcx;
        [RotMat.M]=MatrixRotate(Tree.Mbr(MN).Vec,PreRotate.M,rot);
        Cube.M(nstart:nfin,start:fin)=RotMat.M;
        PreRotate.K=Connect.Neg.Kcx;
        [RotMat.K]=MatrixRotate(Tree.Mbr(MN).Vec,PreRotate.K,rot);
        Cube.K(nstart:nfin,start:fin)=RotMat.K;
        %corner
        mstart = Tree.Mbr(i).Cnx(j).start;
        mfin = mstart+size(Connect.Pos.Mcx)-1;
        PreRotate.M=Connect.Pos.Mcx;
        [RotMat.M]=MatrixRotate(Tree.Mbr(MN).Vec,PreRotate.M,rot);
        Cube.M(mstart:mfin,start:fin)=RotMat.M;
        PreRotate.K=Connect.Pos.Kcx;
        [RotMat.K]=MatrixRotate(Tree.Mbr(MN).Vec,PreRotate.K,rot);
        Cube.K(mstart:mfin,start:fin)=RotMat.K;
    end
end
```

```
        end
    end
end

[Graphic.M,Graphic.K,Graphic.mdof]=fe_mknl(Model);

l=length(Graphic.M);
for c=1:l
    for r=1:l
        if abs(Graphic.M(r,c))<10^(-9);
            Graphic.M(r,c)=0;
        end
        if abs(Graphic.K(r,c))<10^(-9);
            Graphic.K(r,c)=0;
        end
    end
end

l=length(Cube.M);

for c=1:l
    for r=1:l
        if abs(Cube.M(r,c))<10^(-9);
            Cube.M(r,c)=0;
        end
        if abs(Cube.K(r,c))<10^(-9);
            Cube.K(r,c)=0;
        end
    end
end

for r=1:l
    for c=r:l
        if Cube.M(r,c)~=0;
           Cube.M(c,r)=Cube.M(r,c);
        end
        if Cube.K(r,c)~=0;
           Cube.K(c,r)=Cube.K(r,c);
        end
    end
end
```

ModelBuild.m

```
def = fe_eig(Model,[5 20 1 11])

% generate a fairly complex plot as an illustration of feplot
cf=feplot; cf.model=Model;cf.def=def;
```

Spymatrix.m

```
figure('Name','Full Matrices (spy cmd) formed by Model Tree/Matrix Comp','NumberTitle','off');
hold on;
subplot(1,2,1)
spy(Cube.M);; title('Analytical.M - Mass');
subplot(1,2,2)
spy(Cube.K); title('Analytical.K - Stiffness');
hold off
```

```
figure('Name','Full Matrices (spy cmd) formed by Data Entry','NumberTitle','off');
hold on;
subplot(1,2,1)
spy(Graphic.M);; title('Graphic.M - Mass');
subplot(1,2,2)
spy(Graphic.K); title('Graphic.K - Stiffness');
hold off
```

unitvec.m

```
function [vprime,Theta]=unitvec(transvec,v,TreeStart,ThetaRot)
%v=[1,1,1]
%TreeStart=[0,0,0];
%transvec=[-1,1,1];
Vector1=[1,0,0];
Vector2=transvec/norm(transvec);

  % Calculate axis vector
    crossprod = cross(Vector1,Vector2);
    dotprod = dot(Vector1,Vector2);
    % Calculate norm of axis vector
    normal = norm(crossprod);

    c2=cos(ThetaRot);
    s2=sin(ThetaRot);
        if abs(c2)<10^(-9);
            c2=0;
        end
        if abs(s2)<10^(-9);
            s2=0;
        end
    R1 =  [   1    0   0;
              0   c2  s2;
              0   -s2  c2]   ;


    if normal==0,
        if dotprod == 1
          vprime = v*R1+TreeStart;
          A = [];
          Theta = [];
        return
        else if dotprod == -1
          A = [0 -1 0 ];
            end
        end
    else
        A = crossprod/normal;
    end
    Theta = -acos(dotprod/(norm(Vector1)*norm(Vector2)));

    c = cos(Theta);
    s = sin(Theta);
        if abs(c)<10^(-9);
            c=0;
        end
        if abs(s)<10^(-9);
            s=0;
        end
    c2=cos(ThetaRot);
    s2=sin(ThetaRot);
        if abs(c2)<10^(-9);
            c2=0;
        end
```

```
        if abs(s2)<10^(-9);
            s2=0;
        end
    % Rodrigues' Rotation Formula using Theta and v, not Euler angles
    R2 =  [ (c+A(1)^2*(1-c))          (A(1)*A(2)*(1-c)-A(3)*s)     (A(2)*s+A(1)*A(3)*(1-c));
          (A(3)*s+A(1)*A(2)*(1-c))      (c+A(2)^2*(1-c))         (-A(1)*s+A(2)*A(3)*(1-c));
          (-A(2)*s+A(1)*A(3)*(1-c))   (A(1)*s+A(2)*A(3)*(1-c))     (c+A(3)^2*(1-c))];


    v;
    vprime = v*R2*R1+TreeStart;
```

MatrixRotate.m

```
function [RotMat]=unitvec(transvec,matrix,ThetaRot)
%v=[1,1,1]
%TreeStart=[0,0,0];
%transvec=[-1,1,1];
Vector1=[1,0,0];
Vector2=transvec/norm(transvec);

  % Calculate axis vector
    crossprod = cross(Vector1,Vector2);
    dotprod = dot(Vector1,Vector2);
    % Calculate norm of axis vector
    normal = norm(crossprod);

    c2=cos(ThetaRot);
    s2=sin(ThetaRot);
        if abs(c2)<10^(-9);
            c2=0;
        end
        if abs(s2)<10^(-9);
            s2=0;
        end
    R1 =  [   1    0   0;
              0   c2  s2;
              0   -s2  c2]   ;


    if normal==0,
        if dotprod == 1
            Lambda=sparse([R1 zeros(3,3);zeros(3,3) R1]);
            LambdaSize=size(matrix)/6;
            for q=1:LambdaSize
                D((q-1)*6+1:q*6,(q-1)*6+1:q*6)=Lambda;
            end
            RotMat = D'*matrix*D;

            A = [];
            Theta = [];
        return
        else if dotprod == -1
          A = [0 -1 0 ];
            end
        end
    else
        A = crossprod/normal;
    end
    Theta = -acos(dotprod/(norm(Vector1)*norm(Vector2)));

    c = cos(Theta);
    s = sin(Theta);
        if abs(c)<10^(-9);
            c=0;
```

```
    end
    if abs(s)<10^(-9);
        s=0;
    end
c2=cos(ThetaRot);
s2=sin(ThetaRot);
    if abs(c2)<10^(-9);
        c2=0;
    end
    if abs(s2)<10^(-9);
        s2=0;
    end
% Rodrigues' Rotation Formula using Theta and v, not Euler angles
R2 =  [ (c+A(1)^2*(1-c))          (A(1)*A(2)*(1-c)-A(3)*s)     (A(2)*s+A(1)*A(3)*(1-c));
      (A(3)*s+A(1)*A(2)*(1-c))      (c+A(2)^2*(1-c))         (-A(1)*s+A(2)*A(3)*(1-c));
      (-A(2)*s+A(1)*A(3)*(1-c))    (A(1)*s+A(2)*A(3)*(1-c))     (c+A(3)^2*(1-c))];


R=R2*R1;
Lambda=sparse([R zeros(3,3);zeros(3,3) R]);
LambdaSize=size(matrix)/6;
for q=1:LambdaSize
    D((q-1)*6+1:q*6,(q-1)*6+1:q*6)=Lambda;
end
RotMat = D'*matrix*D;
```

Bibliography

1. Chandrupatla, Tirupathi R. and Ashok D Belegundu. *Introduction to Finite Elements in Engineering, Second Edition*. Upper Saddle Rive, NJ: Prentice Hall, 1991.

2. Cook, Robert D. and others. *Concepts and Applications of Finite Element Analysis, Third Edition.* New York, NY: John Wiley and Sons, 1989.

3. Craig, Roy R., Jr. "A Brief Tutorial on Substructure Analysis and Testing," *ASE-EM Department, University of Texas at Austin*.

4. Craig, R. R., Jr. and M. C. C. Bampton. "Coupling of substructures for dynamic analysis," *AIAA Journal*, 6: 1313-1319 (1968).

5. Leiven, N. A. F. "Structural Dynamics Toolbox Primer," *Department of Aerospace Engineering, University of Bristol.*

6. Balmes, Etienne. "Superelement Representation of a Model with Frequency Dependent Properties," *ISMA 21, Leuven*, September 1996.

7. Balmes, Etienne. "Model reduction for systems with frequency dependent damping properties," *International Modal Analysis Conference*. 223--229, 1997.

8. Cebrowski, Arthur K., Director of Force Transformation, Office of the Secretary of Defense. "Statement before the Subcommittee on Strategic Forces," Armed Services Committee, United States Senate, 25 March 2004.

9. Jilla, Cyrus D. and Dr. David W. Miller. "Satellite Design: Past, Present, and Future," *International Journal of Small Satellite Engineering*, (12 Feb 1997).

10. Strunce, Robert, and others. "Responsive Space's Spacecraft Design Tool (SDT)," *AIAA, 4th Responsive Space Conference,* 2006.

11. Knight, Don. "Concept of Operations for Operationally Responsive Space"*, AIAA, 4th Responsive Space Conference,* 2006.

12. Correll, Randall R. "Responsive Space: Transforming U.S. Space Capabilities," Washington Roundtable on Science and Public Policy, Washington, DC, August, 2004.

13. Wie, Bong. *Space Vehicle Dynamics and Control*. Tempe, AZ: American Institute of Aeronautics and Astronautics, Inc. 1998.

14. Wegner, Peter M., AFRL JWS Office Lead, AFRL Space Vehicles Directorate, "Joint Warfighting Space, Vision and Enabling Technologies", Plug-and-Play Thermal Workshop, 24 May 2005.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From – To)*
22/03/2007	**Master's Thesis**	Apr 06 – Mar 07

4. TITLE AND SUBTITLE

Quick-Turn Finite Element Analysis for Plug-and-Play Satellite Structures

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Naff, Jeffrey E., Captain, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)
Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GA/ENY/07-M15

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AFRL/VSSV
Integrated Structural Systems
3550 Aberdeen Ave SE
Kirtland AFB, NM 87117

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Plug-and-play (PnP) satellite construction is a key component of the US Air Force Operational Responsive Space (ORS) effort. The goal of ORS is to provide mission specific satellite support by configuring and launching a satellite to a selected orbit within days of the request. One major challenge during the time limited process is to accurately predict the response of the satellite to harmonic loads that occur during launch and satellite operation. Given the time limitations, constructing finite element (FE) models by traditional methods is not currently a viable option for the ORS timeline. By implementing an approach for rapid FE model creation, we can significantly reduce the timeline from weeks to hours. The advantages to our approach include simplification of model creation, ease of design modifications, and significant reduction in the FE model creation timeline; all lending this approach for utilization within the ORS acquisition cycle.

15. SUBJECT TERMS
Finite Element Analysis, Operational Responsive Space, Plug-and-Play, Nodal Segregation

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
REPORT	**ABSTRACT**	**c. THIS PAGE**			Eric D. Swenson, Maj, USAF (ENY)
U	**U**	**U**	**UU**	118	**19b. TELEPHONE NUMBER** *(Include area code)* (937) 255-3636, ext 7479; e-mail: eric.swenson@afit.edu